Faculdade de Engenharia da Universidade do Porto

FEUP

Protótipo de Procura Semântica aplicado a um Motor de Busca Documental

Luís José Frederico Barreto

Dissertação realizada no âmbito do
Mestrado Integrado em Engenharia Electrotécnica e de Computadores
Major Automação

Orientador: Prof. Dr. Eugénio Oliveira
Co-orientador: Engs. António Castro e Luís Sarmento

Maio de 2008

Resumo

Este projecto tem como finalidade a concepção de um protótipo de procura semântica aplicado a um motor de busca documental (PPSAMBD) realizado no âmbito do Mestrado Integrado em Engenharia Electrotécnica e de Computadores, ano lectivo 2008/09.

A procura semântica faz parte, hoje em dia, das funcionalidades dos motores de busca actuais, enquadrados na terceira Geração da Web, onde as informações são recuperadas com base na semântica das informações contidas nos documentos, ou seja, os dados são recuperados com base em informações estruturadas semanticamente.

Actualmente existem alguns destes motores como o exemplo do "RAPOSA" cuja Extracção de Informação e a fase de recuperação tenta oferecer uma contínua cadeia de processamento on-line, da questão à resposta.

O PPSAMBD consiste, basicamente, numa tentativa de oferecer uma funcionalidade extra ao motor de busca Portal DOV, desenvolvido em 2007, no âmbito do projecto "Motor de pesquisa sobre base de dados documental do Portal DOV", realizado na TAP Portugal, pelo aluno Luís Barbosa.

Este protótipo procura fazer extracção semântica a partir dum ficheiro de texto com cerca de 3481 tokens, citando todas as ocorrências das Unidades Terminológicas (UTs) e as suas respectivas co-ocorrências com outras UTs, culminando numa lista indexada de UTs co-ocorrentes guardados num outro ficheiro de texto, onde serão posteriormente objectos de pesquisa. Para além da extracção dos termos, o sistema é submetido a uma avaliação, onde se tenta salientar a qualidade do mesmo, baseado em duas medidas básicas de qualidade na extracção de termos, que são a Sensibilidade (Recall) e a Precisão. Ambos são definidos em termos de relevância, isto é, quais dos termos encontrados em todo o documento são relevantes, onde o conceito de relevância varia fortemente em função da aplicação em causa.

Apesar de não se ter implementado o módulo de integração deste protótipo com o motor de pesquisa documental do Portal DOV, os resultados alcançados relativamente à extracção de termos, são bastante satisfatórios do ponto de vista da qualidade de extracção, conseguindo com isso, um passo importante na direcção dum modelo de procura semântica.

Agradecimentos

Agradeço a todas as pessoas que contribuíram directa ou indirectamente para a realização desta tese de mestrado sem o qual este trabalho no teria sido possível. Ao meu orientador, o Professor Dr. Eugénio da Costa Oliveira, e os co-orientadores, o Engenheiro António Jesus Monteiro de Castro, pela orientação e revisão da escrita da tese e documentação disponibilizada, e apoio constante, e em especial ao Engenheiro Luís Sarmento pela atenção dispensada ao longo do período de trabalho, bem como a troca de ideias e sugestões preciosas, que por vezes foram as soluções de alguns problemas que, obviamente, faziam parte do trabalho, e claro pela documentação disponibilizada e o apoio constante.

Agradeço ainda aos amigos e colegas de trabalho que sempre fizeram aquela pergunta de coragem "...então Luís o trabalho está a correr bem?", à minha família que apesar da distância que nos separa, deu-me, sempre, aquele apoio moral e psicológico, tornando isto a minha força e vontade de terminar, apesar de todas as barreiras e dificuldades encontradas pelo caminho.

Palavras-Chave

Termos, Extractor de termos, Unidade Terminológica, Pesquisa semântica, Motor de busca, Motor de busca documental, Web Semântica.

Índice

Lista de figuras

Lista de tabelas

Abreviaturas e Símbolos

Lista de abreviaturas (ordenadas por ordem alfabética)

DEEC	Departamento de Engenharia Electrotécnica e de Computadores
DT	Detector de Termos
F	Medida F
FEUP	Faculdade de Engenharia da Universidade do Porto
IR	Informação Relevante
INR	Informação Não Relevante
LIACC	Laboratório de Inteligência Artificial e Ciências de Computadores
P	Precisão
PERL	Practical Extraction and Report Language
PPSAMBD	Protótipo de Procura Semântica aplicado a um Motor de Busca Documental
R	Recall
RDF	Resource Description Framework
TAP	Transportes Aéreos Portugueses
UT	Unidade Terminológica
XML	eXtensible Markup Language
W3C	World Wide Web Consortium

Capítulo 1

Introdução

A TAP Portugal tem uma aplicação Web que os tripulantes e outros utilizadores usam, para ter acesso a um conjunto vasto de informação [1]. Esta aplicação contém um motor de busca que pesquisa informação existente no conteúdo dos documentos (PDF's) publicados pelos vários serviços da empresa. Estes documentos estão guardados em formato binário numa base de dados Oracle que indexa os seus conteúdos.

Contudo a pesquisa actual é feita por "keywords". Neste documento é desenvolvido um protótipo de pesquisa semântica (por oposição a pesquisa por "keywords") para que seja possível fazer pesquisas do género: qual é o limite máximo de bagagem de mão?

No âmbito dos Sistemas de Recuperação de Informação, a organização e a recuperação de informações sempre estiveram condicionadas à tecnologia associada. Actualmente, bases de dados de todos os tipos têm proliferado com a disponibilização de informações em rede e, principalmente, na Web. A recuperação dos conteúdos informativos ainda não é realizada de forma satisfatória, devido à falta de ferramentas de acesso adequadas, que viabilizem, por exemplo, o controlo terminológico.

Para garantir esta precisão, verifica-se a necessidade de ferramentas taxonómicas e terminológicas para o tratamento semântico de informações contidas em bases de dados, viabilizando, entre outros processos, a integração de informações como auxílio ao desenvolvimento de pesquisa em domínios de conhecimento. Ferramentas semânticas como ontologias precisam ser construídas em meio informatizado, abarcando domínios de estudos e pesquisa em língua portuguesa, para serem utilizadas em bibliotecas digitais e virtuais, em sistemas para gestão de conhecimento, para viabilizarem processos de integração de informações entre pesquisadores, como auxílio para as ferramentas de busca de um modo geral, e, principalmente, como um instrumento para a melhoria do tratamento e da recuperação de informação na rede.

1.1 - Motivação

- Distinguir novos termos:

  - Para a construção de um **domínio ontológico específico** a partir de uns documentos não estruturados;

- Levar o Portal DOV da TAP a um novo nível:

  - Inclusão de Pesquisa Semântica no sistema;

  - Introduzir um novo conceito de pesquisa mais inteligente que retribui resultados mais satisfatórios aos utilizadores.

1.2 - Objectivo

Concepção dum Protótipo de Procura Semântica Aplicado a um Motor de Busca Documental (PPSAMBD) suportado na linguagem de programação PERL (Practical Extraction and Report Language)

1.3 - Estrutura do trabalho

Estruturalmente este projecto encontra-se dividido em 4 capítulos, focando o interesse no capítulo 3, onde entramos na parte do desenvolvimento, propriamente dito, da procura semântica. Contudo, os outros capítulos constituem o "invólucro", ou seja, partes introdutórias tanto do projecto em si, como da matéria em questão, buscando com isso situar e contextualizar o leitor num âmbito geral.

- **Capitulo 1** – faz uma pequena introdução ao projecto, citando com isso o porquê desta iniciativa, bem como a motivação e os objectivos.

- **Capítulo 2** – constituído pelo Estado da Arte desta matéria, visa relacionar o passado e o presente, ou seja, o que já se fez e o que está a ser feito relativamente às pesquisas semânticas a nível da Web.

- **Capítulo 3** – neste capítulo desenvolve-se o projecto de procura semântica, tentando com isso um motor de busca simples e robusto que consiga extrair termos de acordo com a ontologia definida para o nosso domínio (manuais de voos da TAP).

- **Capítulo 4** – composto pela conclusão do trabalho, visa não só realçar os aspectos tratados no projecto, mas também, uma projecção do futuro e o que poderá ser feito.

Capítulo 2

Estado da arte

2.1. Introdução

A Internet é uma grande conquista tecnológica com um número crescente de utilizadores e fonte de informações. Entretanto, o aumento da complexidade na Web afecta directamente os utilizadores deixando-os responsáveis por controlar acesso, extracção, interpretação e manutenção de informação. A facilidade de se construir páginas HTML possibilita que todos os dias novos documentos sejam disponibilizados na Web sem controlo de conteúdo ou mesmo controlo de permanência, permitindo com isso que os mesmos documentos sejam alterados com facilidade e até eliminados da rede, pelos seus responsáveis.

Os motores de busca surgiram para facilitar o acesso a esse grande contingente de informação disponível, de maneira rápida e confortável, dando ao utilizador a facilidade de encontrar a informação desejada entre tantas outras. Os motores de busca processam informações contidas nos documentos durante o processo de recuperação e comparam com a consulta requerida pelo utilizador, retornando uma lista dos documentos que contêm similaridade com o assunto desejado. Essas informações são classificadas como informações intrínsecas e informações extrínsecas. Informações intrínsecas são contidas dentro dos documentos que são sujeitos a análises pelos motores de busca. Por exemplo, a ocorrência de um determinado termo num documento e sua localização no texto são informações levadas em consideração durante a elaboração do ranking dos resultados, onde a ordem de listagem é no sentido decrescente de relevância, enquanto que informações extrínsecas são obtidas a partir dos demais documentos contidos na colecção, estrutura de links (Link analysis) ou popularidade de um documento em relação a outros.

Sendo assim podemos classificar os motores de busca da seguinte maneira em termos de evolução:

- Primeira Geração. Durante o processo de recuperação da informação utilizam-se basicamente informações intrínsecas aos documentos.
- Segunda Geração. Para além das informações intrínsecas, utiliza também informações extrínsecas no processo de recuperação de informações.
- Terceira Geração. Recuperam informações com base na semântica das informações contidas nos documentos, ou seja, recuperam dados com base em informações estruturadas semanticamente.

Com as novas necessidades introduzidas pelas motores de busca de terceira geração e outras aplicações, como por exemplo ter conhecimento semântico do conteúdo do documento, ou saber do que se trata uma imagem inserida numa página da Web, surge a necessidade de estruturação dos documentos disponibilizados. A Web Semântica traz uma proposta interessante, juntamente com mecanismos funcionais numa tentativa de resolver o problema.

A **Web semântica** é uma extensão da Web actual, que permitirá aos computadores e humanos trabalharem em cooperação. A Web semântica interliga significados de palavras e, neste âmbito, tem como finalidade conseguir atribuir um significado (sentido) aos conteúdos publicados na Internet de modo que seja perceptível tanto pelo humano como pelo computador.

A ideia da Web Semântica surgiu em 2001, quando Tim Berners-Lee, James Hendler e Ora Lassila publicaram um artigo na revista Scientific American, intitulado: “Web Semântica: um novo formato de conteúdo para a Web que tem significado para computadores vai iniciar uma revolução de novas possibilidades.” [2].

O objectivo principal da Web semântica não é, pelo menos para já, treinar as máquinas para que se comportem como pessoas, mas sim desenvolver tecnologias e linguagens que tornem a informação legível para as máquinas. A finalidade passa pelo desenvolvimento de um modelo tecnológico que permita a partilha global de conhecimento assistido por máquinas. A integração das linguagens ou tecnologias eXtensible Markup Language (XML), Resource Description Framework (RDF), arquitecturas de metadados, ontologias, agentes computacionais, entre outras, favorecerão o aparecimento de serviços Web que garantam a interoperabilidade e cooperação.

Ultimamente tem-se associado Web Semântica a Web 3.0, como um próximo movimento da Internet depois da Web 2.0 que já inicia seu crescimento.

A figura 1 ilustra os diferentes módulos/camadas duma arquitectura para a Web Semântica. A W3C definiu uma arquitectura para padronizar o desenvolvimento da Web Semântica e estas camadas desempenham funções que interagem umas com as outras.

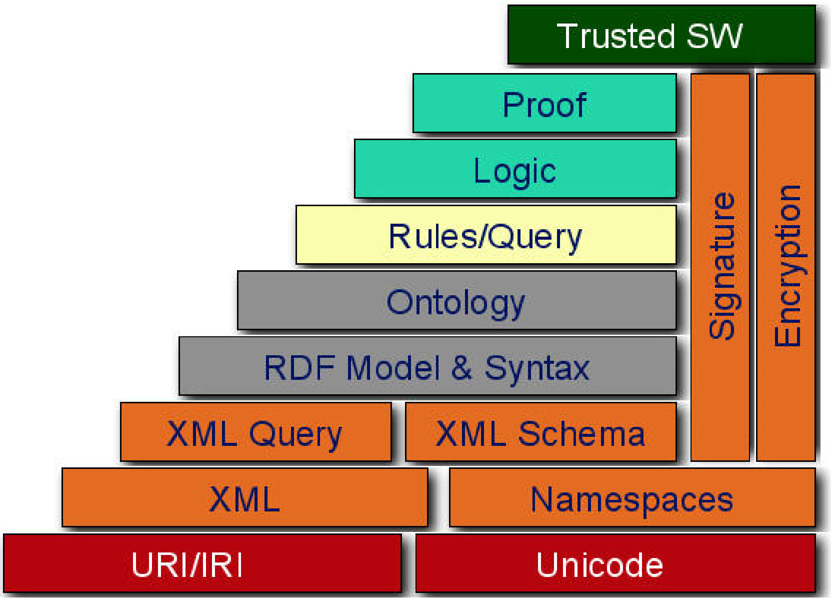

Fig. 1 - Arquitectura da Web semântica

Com a finalidade de alinhar as comunidades científicas que estão envolvidas no progresso da Web, a W3C liderada por Tim Bernes-Lee projectou esta arquitectura e definiu as camadas da seguinte maneira:

URI – Universal Resource Identifier - é um formato que serve como um meio de identificação abstracta ou física de um recurso na Web. Uniform Resource Locator (URL) refere-se a um subconjunto de URI que identifica recursos através de uma representação dos seus principais mecanismos de acessos.

Xml – eXtensible Markup Language. Uma linguagem framework que é usada para definir quase todas as novas linguagens que são usadas para trocar dados na Web.

Xml-Schema - Uma linguagem usada para definir a estrutura de linguagens.

RDF - Resource Description Framework. Uma linguagem flexível capaz de descrever todo o tipo de informações e metadados.

RDF-Schema – Um framework que proporciona um meio para especificar vocabulários básicos para aplicações específicas RDF.

Ontology – São linguagens usadas para definir vocabulários e estabelecer o uso de palavras e termos no contexto de um vocabulário específico. RDF-Schema é um framework para construção de ontologias e é usado por frameworks de ontologias muito mais avançados.

Logic e Proof – Raciocínio lógico é usado para estabelecer a consistência e correcção de conjuntos de dados e inferir conclusões que não são explicitamente declarados, mas são requeridos pela consistência ou com um conjunto de dados conhecidos.

Web semântica depende da combinação das seguintes tecnologias:

- **Metadados**: permite que páginas da Web possam demonstrar o significado sobre suas próprias informações. Por exemplo, numa página Web pessoal, metadados podem identificar nome, cargo, filiação, estudos, publicação, etc.
- **Ontologias**: descrevem os principais conceitos de um domínio e os seus relacionamentos. Por exemplo, uma ontologia pode conter conceitos como pessoas, professores, cursos, e seus relacionamentos.
- **Raciocínio lógico**: Neste caso torna-se possível tirar conclusões a partir das combinações dos dados (metadados) com ontologias.

2.2. A Web actual

Através do uso de portais temos um método de aplicar conhecimento mesmo que ainda através do esforço humano e desta maneira manter uma adequação categorizando as páginas e suas hiperligações associadas. Estes portais são, hoje, de grande importância pois o homem ainda é o grande gestor desta ferramenta introduzindo conhecimento à informação e ainda será por alguns anos.

Os motores de busca, são ferramentas que dão atenção especial aos metadados nas páginas da Web, pois estes os adicionam aos seus índices nas suas bases de dados. Estes metadados são de grande importância para as mais avançadas ferramentas de buscas como o Google, que não só avalia a ocorrência de palavras-chave numa página, mas também os números de hiperligações existentes para outra página na Web.

Em qualquer solicitação a estes motores de busca é retornado informações que não se relacionam com o contexto ou ainda que simplesmente não fazem parte do significado desta informação. Diante destes factores, a Web necessita de uma tecnologia que possa catalogar e classificar os seus próprios conteúdos e possibilitar assim que sejam desenvolvidos agentes de software que poderia explorar a informação nas páginas da Web. O que facilitaria o uso da Web sem ter de depender destas ferramentas de buscas ou mesmos de portais, tornando a informação adaptada à particularidade de cada utilizador. Estes agentes podem ainda aprender e responder as informações de outros agentes actuantes na Web com a utilização de conceitos de inteligência computacional.

2.3. Motores de busca actuais

Actualmente existem inúmeros motores de busca que realizam as suas pesquisas de forma tradicional através de palavras - chave, tratando as páginas Web como “caixas de palavras” e indexando o seu conteúdo sem compreender o seu significado.

Contudo alguns destes Motores estão a inverter esta tendência, como por exemplo “Tropes” [3] que pode catalogar quantos adjectivos, verbos ou substantivos foram usados e compará-los. O **Tropes** quantifica e qualifica as palavras usadas em português e castelhano.

Para além do Tropes, o **Powerset** [4] pretende recorrer a tecnologia proprietária e a tecnologia licenciada pela PARC (subsidiária da Xerox) para criar uma representação semântica ao analisar sintacticamente cada frase para dela extrair o seu significado. Para já, o índice do Powerset é muito limitado, tendo apenas alguns milhões de páginas da Wikipedia e da Freebase da Metaweb Technologies, uma estrutura de base de dados para informação baseada em Web.

Um outro caso de motores actuais, temos o **RAPOSA** [6] que ao contrário de muitos outros sistemas que claramente separam a fase de Extracção de Informação (onde os factos são extraídos das bases de textos e armazenados numa base de dados intermédia) e a fase de recuperação (onde, dada uma questão, o sistema recupera respostas candidatas da base de dados e gera uma resposta), o RAPOSA tenta oferecer uma contínua cadeia de processamento on-line, da questão à resposta.

Para além destes a Google também já apresentou as novas funcionalidades de pesquisa semântica que está em fase experimental.

Contudo estas novas tecnologias ainda enfrentam alguns problemas devido às características da Web, ou seja:

- **A Web é distribuída**: sendo a Web não constituído por uma autoridade central, isto constitui um entrave uma vez que é um produto composto por várias individualidades, a falta de um controlo centralizado apresenta muitos desafios para dar significado às informações.

- **A Web é dinâmica**: O ritmo acelerado das mudanças das informações na Web representa um desafio a mais para qualquer tentativa de criar vocabulários padrão e fornecer semântica formal. Como o entendimento de um dado domínio muda, tanto o vocabulário quanto a semântica podem ser refinados.

É importante que tais modificações não alterem negativamente os significados de conteúdo existentes.

- **A Web é abrangente**: segundo a *NetCraft* [5], actualmente existem cerca de 236 milhões de sites em funcionamento por todo o globo. Diante disso, a implementação de um sistema de reconhecimento semântico torna numa tarefa árdua.

Capítulo 3

Procura Semântica

3.1 Definição

Termos e vocabulário geral: A extracção de termos significa identificar os candidatos a partir de um corpo de texto. Termos são uma "designação de um conceito definido numa língua especial por uma expressão linguística. O termo pode ser constituído por uma ou mais palavras." [7]. Esta definição refere-se a uma diferença entre linguagem geral e linguagem especial, atribuindo condições especiais para a área linguística. Contudo, isto não constitui uma limitação aos sistemas de extracção de termos, como se verá em seguida.

Extracção e reconhecimento de termo: Existem duas fases a ser distinguidos, sendo ambas por vezes chamadas de extracção de termos: O primeiro deles é a extracção de termos propriamente dita, também chamado de aquisição de termos [8], ou seja, a identificação de termos candidatos no corpo de texto, e o segundo é um reconhecimento de termos [8], ou seja, comparação do resultado da extracção com uma determinada fonte, de forma a identificar termos conhecidos/desconhecidos. Contudo há aplicações de extracção de termos que dispensa o reconhecimento dos termos.

Termos simples e termos compostos: Há uma diferença entre termos constituídos por uma palavra e termos constituídos por várias palavras; um termo composto é um termo constituído por várias palavras mas com uma unidade semântica. A maioria dos termos, cujo sistema de extracção de termos é suposto encontrar, como por exemplo cabina dos tripulantes de voo, são termos compostos, sendo termos simples raramente úteis.

3.2 Aplicação

Na avaliação dos sistemas de extracção de termos, os critérios de avaliação devem depender do propósito da extracção desses termos, conforme especificado na norma ISO 9126 [9] para adequação. Por isso, antes duma análise mais atenta às aplicações cuja extracção se destina é necessário um planeamento da avaliação. Torna-se rapidamente nítido que há muitos diferentes propósitos para as quais estes sistemas podem ser utilizados, impondo estes propósitos exigências diferentes a estes sistemas.

Terminologia

A aplicação da extracção de termos consiste na identificação dos termos candidatos. A Terminologia teve sempre uma forte componente normativa, mas por vezes é baseada em considerações empíricas. Um caso especial nesta aplicação é os sistemas que verificam linguagem controlada. Estes sistemas têm maior facilidade no reconhecimento de termos uma vez que a sua tarefa é identificar palavras que não pertencem à língua geral, nem a um domínio especial de uma língua ou até mesmo empresarial. A fim de comparar esses termos controlados, os mesmos devem ser primeiramente extraídos [10]. As principais características desta aplicação em relação à extracção de termos são:

- Identificar um subconjunto específico do vocabulário de um texto, e
- Restringi-las a uma linguagem dum domínio específico. A maioria das funcionalidades da extracção de termos aqui é monolingue, e, naturalmente, deve abranger termos simples e termos compostos; A extracção de termos bilingue é também usada. O reconhecimento de termos é essencial para aplicações de linguagem controlada.

Informação recuperada

O objectivo da extracção de termos na recuperação de informação é diferente novamente. O objectivo aqui é ter uma visão geral sobre um vocabulário pesquisável de uma aplicação (assumindo que se trata de um motor de recuperação especial e não de uma ferramenta de pesquisa global de internet), que é idêntica para a determinação dos tópicos pesquisáveis. A extracção de termos, neste contexto, é um primeiro passo no sentido da definição de recursos linguísticos para a expansão da query, tradução da query, a construção de ontologia, etc. O requisito básico para a extracção de termos aqui é que os utilizadores devem encontrar todos os tópicos pesquisáveis, se estes forem termos ou palavras do vocabulário em geral. Quaisquer que sejam os dados do texto, estes têm que se tornar num objecto de pesquisa. Portanto, as características desta extracção de termos são:

- A identificação de todo vocabulário pesquisável, sendo gerais ou especiais, conhecidos ou não.

- Identificar as relações entre os conceitos encontrados para o uso em ontologias e sistemas de apoio à pesquisa.

3.3 Tecnologia

Um olhar sobre a tecnologia utilizada para extracção de termos é relevante, uma vez que a selecção da tecnologia tem um impacto directo na qualidade da extracção, e depende do objectivo da extracção. A tecnologia de extracção de termos segue um padrão ao combinar o processamento estatístico com o linguístico [13], e identifica possíveis candidatos, e determina a sua relevância.

Identificação de possíveis candidatos

Cada sistema de extracção de termos deve ter um meio de identificação dos candidatos de termos simples. Independentemente da aplicação a que se destina.

O uso de "*listas de stop*" pode ser aconselhável. O conteúdo das listas de stop pode ser diferente, no entanto, dependendo da aplicação a que se destina. Para a terminologia, todas as palavras do vocabulário geral podem ser interrompidas.

Os sistemas de extracção de termos devem igualmente fornecer identificação de termos compostos em todas as aplicações. Para esta finalidade são usados diferentes técnicas:

- **Filtros linguísticos**, baseados em categoria padrões com uma (NP – orientado) análise sintáctica superficial [12] [13] [14], são adequados para identificação de termos, desde que os termos tenham uma estrutura linguística típica. Mesmo que os termos não sejam adjacentes [15], eles podem ser descritos linguisticamente.
- **Dispositivos estatísticos** para coordenar as palavras dentro dos termos composto [17] geralmente origina problemas na medida em que eles geram significativamente mais ruído do que as abordagens mencionadas e como existem muitas strings num corpo de texto que ocorrem com alguma frequência, alguns filtros são usados para reduzir o número de termos candidatos. O filtro mais simples é o de limitar o comprimento do padrão para duas [16] ou três palavras (mais preposições, determinantes etc.). No entanto, o único caso em que a análise estatística parece ser superior à filtragem linguística [15] é quando há uma selecção sub-óptima dos termos padrões linguísticos.

Possíveis termos candidatos são melhores se forem (linguisticamente) conceitos significativos. Todas as outras propostas apenas adicionam ruído aos avaliadores.

3.4 O extractor de termos

Esta secção descreve o sistema de extracção de termos monolingue, desenvolvido tendo em consideração o que foi descrito nas secções anteriores.

Detector de UTs

Primeiramente o texto de entrada (texto de teste) [18] passa por um pequeno filtro de abreviaturas, linhas vazias, etc, através da função tokenize() (Anexo A.1). Sabendo que os substantivos são muitas vezes considerados termos candidatos, a técnica seguida aqui é de fazer uma estrutura n-gram (Anexo A.2) onde as primeiras e últimas palavras são substantivos ou adjectivos. Para implementar este mecanismo simples utilizou-se um "lookup" de dicionário. Contundo todas as sequências fora dos ciclos n-gram seriam simplesmente ignorados, ou seja qualquer unidade termiónica que começa com palavras que não constam no dicionário jamais seria considerado.

A solução para este problema veio de uma simples observação: aparentemente, é muito mais fácil excluir unidades terminológicas (UT) inválidas do que seleccionar as válidas. Isto vai exactamente na direcção oposta dos algoritmos de extracção que tentam identificar unidades baseadas em descrições linguísticas ou estatísticas. Por exemplo, é possível afirmar com um grande grau de probabilidade de que uma unidade terminológica não vai começar com "com". Contudo não se está a afirmar que não há nenhuma unidade terminológica válida que começa por "com", porque poderá haver um domínio particular onde "com" é, na verdade, uma unidade terminológica válida. O que se está a dizer é que em vez tal ser uma regra lexical básica, é notável na grande maioria dos domínios, e pode, por isso, ser utilizado para excluir, com precisão, candidatos inválidos. O mesmo pode se dizer a cerca de determinadas palavras que nunca podem terminar, ou até mesmo fazer parte de qualquer UT.

A figura 2 mostra-nos um esquema de extracção de UTs onde os blocos **Start**, **Stop**, **Forbidden**, **Texto** e **Uts** são ficheiros de texto (txt).
Dictionary.pl é um ficheiro .pl (Anexo A.5) cujo algumas funcionalidades serão discutidas em seguida.

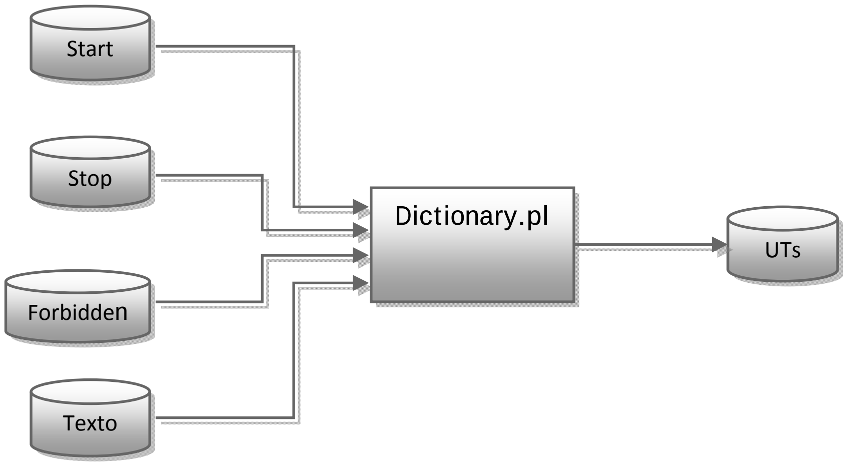

Fig. 2 - Esquema de extracção de UTs

Lista de não-Starts – palavras que nunca podem inicializar uma UT válida, como por exemplo: “a”, “um”, “pelo”, “neste”, “de”, “as”, “seus”, “outros”, etc

Lista de não-Stop - palavras que, nunca podem finalizar uma UT válida, como por exemplo: “até”, “à”, “pelo”, “onde”, “nos”, “as”, “para”, “;”, etc

Lista de proibidos - palavras que, nunca podem ocorrer numa UT válida, como por exemplo: “:”, “neste”, “mesmo”, “frente”, “)”, “para”, “seus”, “noutros”, etc

Texto – o texto é basicamente um extracto do Manual [18].

Dictonary.pl – este ficheiro contem algumas funções que basicamente percorrem um corpo de texto à procura de elementos da lista não-start (função CompareString()). Esta função recebe dois argumentos que lhe são passados, um contendo a string que é retirada do texto e o outro o hash que posteriormente guarda as UTs extraída através da função extair(). O algoritmo de extracção é mostrado na figura 4.

```
sub CompareString($$) {
  my $token_ref = shift;
  my $result_hash = shift;
  my @tokens = @{$token_ref};
  my $max = $#tokens - 1;
  for(my $ii = 0; $ii <= $max; $ii++) {
    if($start{lc($tokens[$ii])} == 1) {
      my $delta = extrai($ii + 1, \@tokens, $result_hash);
      $ii += $delta;
    }
  } ## for

}
```

```
sub extrai($$$) {
  my $n = shift;
  my $tokens_ref = shift;
```

```
  my $result_hash = shift;
  my @tokens = @{$tokens_ref};
  my $item = "";
  my $max = $#tokens;
  for(my $jj = $n; $jj <= $max; $jj++ ) {
       ## Case 1 - A stop token found
    if($stop{lc($tokens[$jj])} == 1) {
       ## case 2 - A start token was given
       if($start{lc($tokens[$jj - 1])} == 1) {
         return $jj - $n;
       } else {
              $$result_hash{$item}++;
         return $jj - $n;
       }
    }
    ## Case 3 - A forbidden token found
    if($forbidden{lc($tokens[$jj])} == 1) {
       return $jj - $n;
    }
    ## Case 4 - End of line found
    if($jj == $max) {
      return $max - $n;
    }
    $item .= " " . $tokens[$jj];
    $item =~ s/^\s+//;  ##remove possible inicial white space
    $item =~ s/^\s+$//; ##remove possible final white space
    $item =~ s/^\n*//;  ##remove possible inicial white line
    $item =~ s/\n*$//;  ##remove possible final white line
  }

}
```

UTs – São as Unidades Terminológicas extraídas a partir do texto de entrada.

Detector de Frases

A extracção de frase processa-se de modo semelhante ao da extracção de Uts, com a diferença de não existir uma lista de elementos proibidos uma vez que todos os tokens são considerados. Para além desta lista, na lista de Stopphrase só entra sinais de pontuação.

A figura 3 mostra-nos um esquema de extracção de frase.

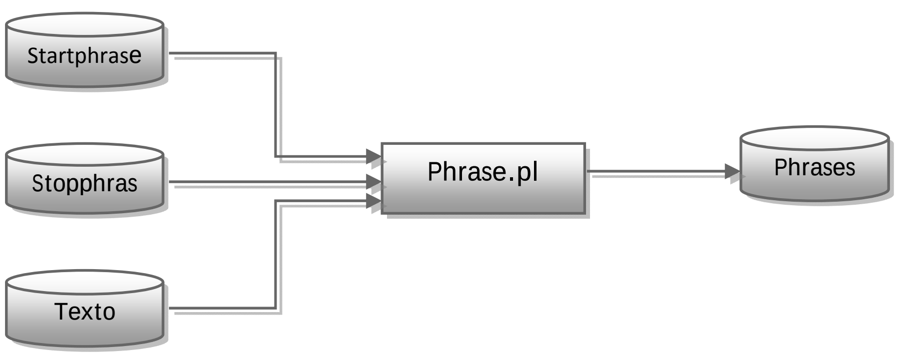

Fig. 3 – Esquema de extracção de frases

Algoritmo de extracção

O algoritmo de extracção consiste, basicamente, em percorrer uma frase e filtrar tudo o que se encontra entre um Start e um Stop, com rejeição dos elementos que se encontram na lista proibida.

O fluxograma da figura 4 dá-nos a percepção do algoritmo de extracção

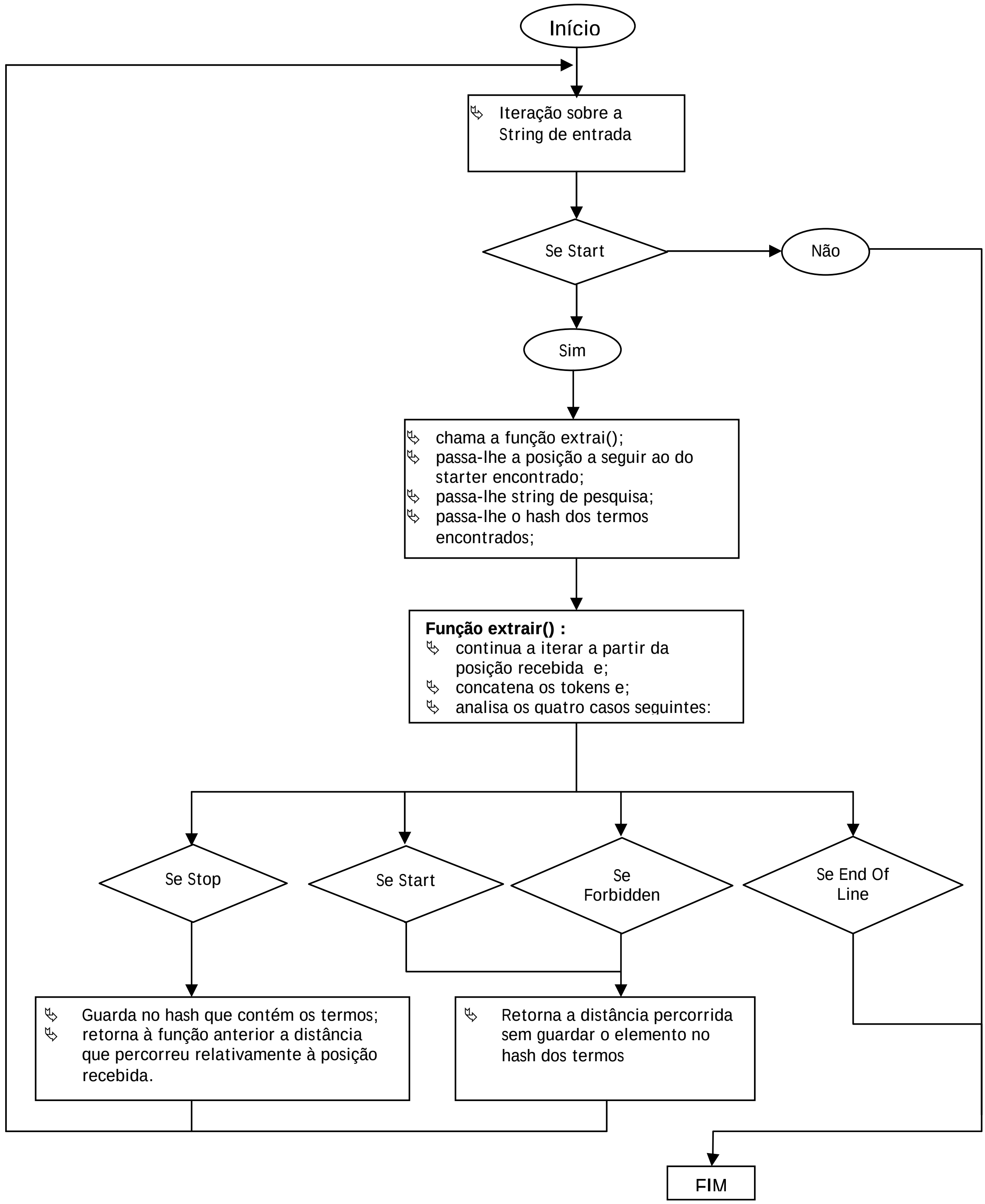

Fig. 4 - Algoritmo de extracção de termos

3.5 Co-ocorrência

Um documento é constituído por frases. Aqui considera-se uma frase como sendo um conjunto de palavras separadas por sinais de pontuação tais como ".", ":" ou ";". Nesta definição, inclui-se títulos dos documentos, títulos das secções, capítulos, etc. O processo de extracção de frase não é muito diferente da técnica utilizada na extracção de termos.

Definição

Diz-se que dois termos co-ocorrem quando ocorrem numa mesma frase. Podemos ver cada frase como sendo um "cesto", ignorando as ordens dos termos, bem como informação gramatical.

A figura 5 mostra-nos o esquema de extracção de listas de UTs co-ocorrentes, bem como as suas indexações.

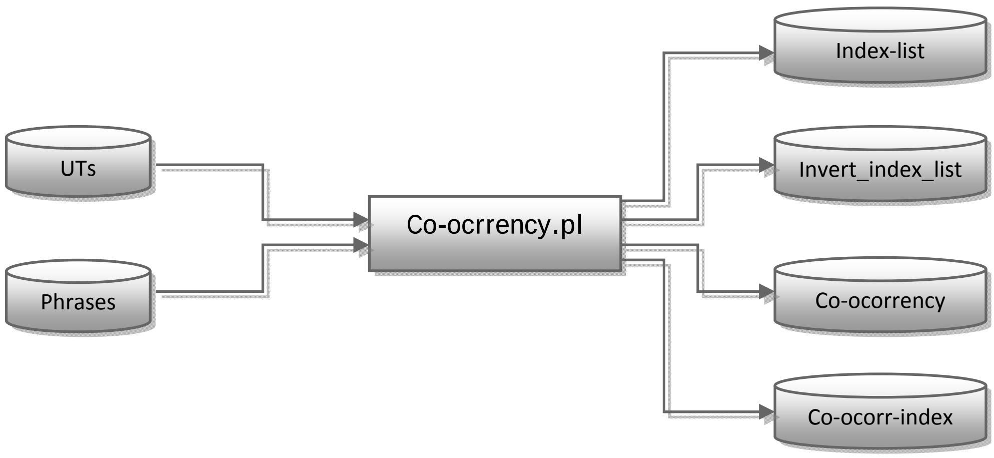

Fig. 5 – Esquema de extracção de listas de UTs co-ocorrentes

UTs – Unidades Terminológicas extraídas através do ficheiro Dictionary.pl

Phrases – Frases extraídas a partir do ficheiro Phrase.pl

Co-ocorrencia.pl – Ficheiro de extracção de UTs co-ocorrentes, e indexação das Uts.

Índex_list – Determina o índice de UT relativamente à frase onde ocorre.

```
sub IndexList() {
  my $filename_out = "index_list.txt";
```

```
my $file_handle = new FileHandle();
if (!$file_handle->open(">$filename_out")) {
  die("Could not open input file '$filename_out'\n");
}
Init();
for(my $ii = 0; $ii <= $#gbl_phrase; $ii++) {
 for(my $jj = 0; $jj <= $#gbl_terms; $jj++) {
        if($gbl_phrase[$ii] =~ /$gbl_terms[$jj]/) {
              push(@{$gbl_term_index{$gbl_terms[$jj]}},$ii);
        } ##if
 }      ##for $jj
} ##for $ii
for(my $ii = 0; $ii <= $#gbl_terms; $ii++) {
 #for(my $jj = 0; $jj <= $#{$gbl_term_index{$gbl_terms[$ii]}};
$jj++) {
        print $file_handle "$gbl_terms[$ii] -
@{$gbl_term_index{$gbl_terms[$ii]}}\n";
 #}
}

}
```

Ex 3.5.1: Grupo - 1 3 10 12 13 15 16 17

Neste exemplo vemos que UT “Grupo” ocorre nas frases cujos índices são 1,3,10, etc.

Invert_index_list – Inverte a “índex_list”, obtendo o conjunto de UTs para cada índice de frase.

```
sub InitInvert() {
  for(my $ii = 0; $ii <= $#gbl_terms; $ii++) {
        for(my $jj = 0; $jj <= $#{$gbl_term_index{$gbl_terms[$ii]}};
$jj++) {
       if(!defined($gbl_index_terms{${$gbl_term_index{$gbl_terms[$ii]}}[$
jj]})) {

  @{$gbl_index_terms{${$gbl_term_index{$gbl_terms[$ii]}}[$jj]}} = ();
                }
        }
  }

}
```

Ex 3.5.2: 3 - Centro Integrador do Grupo   Grupo   apoio jurídico do Centro Integrador do Grupo.

No exemplo 3.5.2 temos que a frase cujo índice é 3 contém as UTs “Centro Integrador do Grupo”, “Grupo” e “apoio jurídico do Centro Integrador do Grupo.”

Co-ocorrency – Cria um lista de termos co-ocorrentes, bem como o números de vez que co-ocorrem.

```
InitInvert();
  for(my $ii = 0; $ii <= $#gbl_terms; $ii++) {
        for(my $jj = 0; $jj <= $#{$gbl_term_index{$gbl_terms[$ii]}};
$jj++) {
       push(@{$gbl_index_terms{${$gbl_term_index{$gbl_terms[$ii]}}[$jj]}}
," $gbl_terms[$ii] ");
       #push(@{$gbl_index_terms{${$gbl_term_index{$gbl_terms[$ii]}}[$jj]}
},"\t");
        }
  }
```

```
  my @keys = sort({$gbl_index_terms{$b} <=> $gbl_index_terms{$a}} keys
%gbl_index_terms);
  foreach my $key (@keys) {
        #printing to the invert_index_list.txt file
    print $file_handle "$key - @{$gbl_index_terms{$key}}\n";
    #making the co-ocorrency between the list elementes
    for(my $ii = 0; $ii <= $#{$gbl_index_terms{$key}}; $ii++) {


               for(my $jj = 0; $jj <= $#{$gbl_index_terms{$key}}; $jj++)
{
                      if($ii != $jj) {
                             #keeping the pair on a hash
                             $pair_list{${$gbl_index_terms{$key}}[$ii]." |
".${$gbl_index_terms{$key}}[$jj]}++;
                             #making the co-ocorrency index list


  push(@{$co_ocorr_index{${$gbl_index_terms{$key}}[$ii]}},${$gbl_index_te
rms{$key}}[$jj]);
                      }
               }## for $jj
        }      ## for $ii
  }
  $file_handle->close();

          }



sub CoOcorrPair() {
 my $filename_out = "co_ocorrency.txt";
  my $file_handle = new FileHandle();
  if (!$file_handle->open(">$filename_out")) {
    die("Could not open output file '$filename_out'\n");
  }
  my @keys = sort({$pair_list{$b} <=> $pair_list{$a}} keys %pair_list);
 foreach my $key (@keys) {

        print $file_handle "$key - $pair_list{$key}\n";
  }
  $file_handle->close();

}
```

Ex 3.5.3: Grupo Missão Principal | Grupo - 4

No exemplo 3.5.3 as UTs “Grupo Missão Principal” e “Grupo” co-ocorrem 4 vezes.

Co-ocorr-index – Cria uma lista de UTs, reportando para cada UT a sua ocorrência com outras UTS e o número de vezes que co-ocorrem.

```
sub CoOcorrIndex() {
 my $filename_out = "co_ocorr_index.txt";
  my $file_handle = new FileHandle();
  if (!$file_handle->open(">$filename_out")) {
    die("Could not open output file '$filename_out'\n");
  }

  my @keys = sort({$co_ocorr_index{$b} <=> $co_ocorr_index{$a}} keys
%co_ocorr_index);
 foreach my $key (@keys) {
        my %temphash = ();
        for(my $ii = 0; $ii <= $#{$co_ocorr_index{$key}}; $ii++) {
               $temphash{${$co_ocorr_index{$key}}[$ii]}++;
        }
        @temp = sort({$temphash{$b} <=> $temphash{$a}} keys %temphash);
        print $file_handle "$key - ";
```

```
        foreach my $key (@temp) {
              print $file_handle "$key($temphash{$key})  ";
        }
        #print $file_handle %temphash;
        print $file_handle "\n";
  }
  $file_handle->close();

}
```

Ex 3.5.4: Grupo Missão Principal - Grupo (4) Recursos Humanos do Grupo Missão Principal (1) Planeamento Estratégico do Grupo Missão Principal (1) Assessoria Jurídica do Grupo Missão Principal (1)

No exemplo 3.5.4 vemos que a UT “Grupo Missão Principal” co-ocorre com vários UTS como por exemplo “Grupo” 4 vezes.

3.6 Avaliação – Critério

Fundamentalmente, a avaliação é sempre uma comparação. A receita EAGLES 7-Step [15] define as dimensões das comparações possíveis. Com base no destino do sistema, um modelo de requisitos deve ser concebido para se poder avaliar os diferentes sistemas utilizando métricas definidas. Na extracção de termos, para além do cumprimento de normas, da documentação de apoio, dos apoios linguísticos, e da apresentação das informações contextuais, o critério central é a qualidade da extracção [19].

Recall e Precisão

A medida básica de qualidade na extracção de termos é a Sensibilidade (Recall) e a Precisão [19] [20]. Ambos são definidos em termos de relevância, i.e. quais dos termos encontradas em todo o documento são relevantes. O problema fundamental com esta abordagem é que é muito difícil definir o quão relevante são os termos. Tal como anteriormente referido, o conceito de relevância varia fortemente em função da aplicação em causa.

Consideremos a figura 6, onde à esquerda temos a Informação Relevante (IR), à direita Informação Não Relevante (INR), a oval representa uma amostra, e a azul os resultados mais correctos.

Precisão pode ser visto como uma medida de exactidão ou de fidelidade, enquanto que o Recall é uma medida de perfeição.

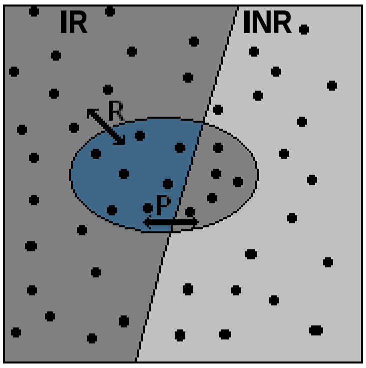

Fig. 6 – Ilustração dos parâmetros de avaliação Precisão e Recall

Na figura 6 podemos ver que quanto mais à esquerda estiver a oval, mais IR conseguimos obter, enquanto que o Recall é dado pelo tamanho da oval, ou seja quanto maior, mais sensível se torna.

A Precisão e o Recall podem ser definidos da seguinte forma:

$$\mathrm{Pr}\,ecis\tilde{a}o = \frac{\left|\{Informação\ \mathrm{Re}\,levante\} \cap \{Amostra\ Considerada\}\right|}{\left|\{Amostra\ Considerada\}\right|} \tag{1}$$

$$\mathrm{Re}\,call = \frac{\left|\{Informação\ \mathrm{Re}\,levante\} \cap \{Amostra\ Considerada\}\right|}{\left|\{Total\ Informação\ \mathrm{Re}\,levante\}\right|} \tag{2}$$

Contudo há um terceiro elemento de avaliação presente, uma medida popular que combina a Precisão e o Recall, que é a média harmónica ponderada da Precisão e do Recall, designado pela medida F ou resultado equilibrado F, que é dado pela seguinte fórmula:

$$F = \frac{2.(R.P)}{(R+P)} \tag{3}$$

3.7 Resultados experimentais

Consideremos o exemplo 3.7.1 em que pegamos num pequeno extracto de texto retirado do Manual de Regulamentação de Voo [18]:

a) Texto de entrada:

"É o tripulante que se encontra devidamente qualificado pela entidade aeronáutica nacional ou pela empresa para, nos equipamentos de wide body, supervisionar e executar o serviço de cabina por forma que seja prestada completa assistência aos passageiros e à tripulação, assegurando o cumprimento das normas de segurança, a fim de lhes garantir conforto e segurança durante o voo."

b) Lista de start:

- §_begin_§
- o
- pela
- nos
- à

c) Lista de stop:

- §_end_§
- ,
- nos
- o
- para
- ou
- por

d) Lista de proibido:

- pelo
- nos
- ,
- se
- que

e) Extracção manual:

- tripulante
- entidade aeronáutica nacional
- equipamento de wide body

- serviço de cabina
- passageiros
- tripulação
- cumprimento das normas de segurança
- conforto e segurança durante o voo

f) Extracção automática:

- É
- tripulante
- entidade aeronáutica nacional
- equipamentos de wide body
- serviço de cabina
- tripulação
- executar
- voo.
- empresa
- cumprimento das normas de segurança

Salienta-se que as listas de Start, Stop e Proibido, também, são parte das listas completas do Start, Stop e Proibido, respectivamente. Mais adiante vamos retomar este exemplo a fim de fazer uma exploração no que diz respeito à avaliação.

Matriz da Co-ocorrência

a) Termos extraídos anteriormente…

a- É
b- tripulante
c- entidade aeronáutica nacional
d- equipamentos de wide body
e- serviço de cabina
f- tripulação
g- executar
h- voo.
i- empresa
j- cumprimento das normas de segurança

b) Frase extraída a partir do texto de entrada:

É o tripulante que se encontra devidamente qualificado pela entidade aeronáutica nacional ou pela empresa para , nos equipamentos de wide body , supervisionar e executar o serviço de cabina por forma que seja prestada completa assistência aos passageiros e à tripulação , assegurando o cumprimento das normas de segurança , a fim de lhes garantir conforto e segurança durante o voo.

Nota: No exemplo 3.7.1 a frase extraída a partir do texto de entrada é coincidente com o texto de entrada, uma vez que o único stop (stop frase) encontrado é o ponto final (.)

Sabendo que cada termo ocorre uma única vez na frase, e sendo apenas uma frase, pode-se ver facilmente na tabela 1 que cada termo co-ocorre apenas uma única vez com os demais.

	a	b	c	d	e	f	g	h	i	j
a	-	1	1	1	1	1	1	1	1	1
b	1	-	1	1	1	1	1	1	1	1
c	1	1	-	1	1	1	1	1	1	1
d	1	1	1	-	1	1	1	1	1	1
e	1	1	1	1	-	1	1	1	1	1
f	1	1	1	1	1	-	1	1	1	1
g	1	1	1	1	1	1	-	1	1	1
h	1	1	1	1	1	1	1	-	1	1
i	1	1	1	1	1	1	1	1	-	1
j	1	1	1	1	1	1	1	1	1	-

Tabela 1 – Matriz da co-ocorrência

Exemplo 3.7.2

Consideremos o caso em que temos a seguinte entrada:

> MACROESTRUTURA
> Conselho de Administração Composto por cinco membros, sendo: um Presidente, três não executivos e um executivo (Administrador-Delegado). Missão Principal: Delibera sobre os assuntos de administração da sociedade nos termos da lei e dos estatutos da empresa
> Administrador-Delegado Missão Principal: Exerce os poderes de gestão da sociedade nos termos da delegação de competências conferida pelo C.A.
> Vice-Presidentes Executivos Missão Principal: São co-responsáveis pela gestão das Áreas de Negócio sob a sua supervisão nos termos da subdelegação de competências do Administrador-Delegado. São nomeados pelo Conselho de Administração sob proposta do Administrador-Delegado.
> Centro Integrador do Grupo
> Planeamento Estratégico do Grupo Missão Principal: Definir a orientação estratégica e a carteira de negócios da TAP e Associadas. Finanças do Grupo Missão Principal: Assegurar a gestão financeira da TAP, maximizando o aproveitamento da posição consolidada dos recursos financeiros do Grupo, alocando-os às Unidades de Negócio, e controlar e estimular a "performance" do Grupo.
> Recursos Humanos do Grupo Missão Principal: Atrair, promover e reter quadros de elevado potencial e apoiar e acompanhar a gestão de Recursos Humanos no Grupo.
> Assessoria Jurídica do Grupo Missão Principal: Apoiar e acompanhar a gestão de Recursos Humanos no Grupo; atrair, promover e reter quadros de elevado potencial; assegurar o apoio jurídico do Centro Integrador do Grupo.

Lista de termos extraídos:

1 - supervisão
2 - Centro Integrador do Grupo
3 - poderes de gestão da sociedade
4 - performance.
5 - gestão financeira da TAP
6 - Grupo

7 - Unidades de Negócio
8 - Presidente
9 - reter quadros de elevado potencial
10 - assuntos de administração da sociedade
11 - Administrador-Delegado.
12 - termos da subdelegação de competências do Administrador-Delegado
13 - Planeamento Estratégico do Grupo Missão Principal
14 - carteira de negócios da TAP e Associadas
15 - Administrador-Delegado Missão Principal
16 - acompanhar
17 - gestão de Recursos Humanos.
18 - Grupo Missão Principal
19 - C.A.
20 - Conselho de Administração Composto
21 - termos da lei e dos estatutos da empresa
22 - gestão de Recursos Humanos
23 - apoio jurídico do Centro Integrador do Grupo.
24 - aproveitamento da posição consolidada dos recursos financeiros do Grupo
25 - controlar e estimular
26 - Vice-Presidentes Executivos Missão Principal
27 - Recursos Humanos do Grupo Missão Principal
28 - executivo Administrador-Delegado
29 - reter quadros de elevado potencial e apoiar e acompanhar
30 - MACROESTRUTURA
31 - Assessoria Jurídica do Grupo Missão Principal

Lista de frases extraídas:

0 - Vice-Presidentes Executivos Missão Principal
1 - Assessoria Jurídica do Grupo Missão Principal
2 - MACROESTRUTURA
3 - apoio jurídico do Centro Integrador do Grupo.
4 - assuntos de administração da sociedade nos termos da lei e dos estatutos da empresa
5 - gestão de Recursos Humanos.
6 - poderes de gestão da sociedade nos termos da delegação de competências conferida pelo C.A.
7 - Presidente , três não executivos e um executivo Administrador-Delegado
8 - Conselho de Administração sob proposta do Administrador-Delegado.
9 - Conselho de Administração Composto por cinco membros , sendo
10 - gestão de Recursos Humanos no Grupo
11 - Administrador-Delegado Missão Principal
12 - Centro Integrador do Grupo
13 - Grupo Missão Principal
14 - orientação estratégica e a carteira de negócios da TAP e Associadas
15 - Recursos Humanos do Grupo Missão Principal
16 - Planeamento Estratégico do Grupo Missão Principal
17 - gestão financeira da TAP , maximizando o aproveitamento da posição consolidada dos recursos financeiros do Grupo , alocando-os às Unidades de Negócio , e controlar e estimular a performance.
18 - gestão das Áreas de Negócio sob a sua supervisão nos termos da subdelegação de competências do Administrador-Delegado

Lista indexada

Lista indexada invertida

16 - Grupo Planeamento Estratégico do Grupo Missão Principal Grupo Missão Principal
15 - Grupo Grupo Missão Principal Recursos Humanos do Grupo Missão Principal
13 - Grupo Grupo Missão Principal
10 - Grupo gestão de Recursos Humanos. gestão de Recursos Humanos
1 - Grupo Grupo Missão Principal Assessoria Jurídica do Grupo Missão Principal
17 - performance. gestão financeira da TAP Grupo Unidades de Negócio aproveitamento da posição consolidada dos recursos financeiros do Grupo controlar e estimular
6 - poderes de gestão da sociedade C.A.
12 - Centro Integrador do Grupo Grupo
3 - Centro Integrador do Grupo Grupo apoio jurídico do Centro Integrador do Grupo.
18 - supervisão termos da subdelegação de competências do Administrador-Delegado

Co-ocorrências

Grupo Missão Principal | Grupo - 4

Grupo | Grupo Missão Principal - 4

gestão de Recursos Humanos | gestão de Recursos Humanos. - 2

gestão de Recursos Humanos. | gestão de Recursos Humanos - 2

Centro Integrador do Grupo | Grupo - 2

Grupo | Centro Integrador do Grupo - 2

controlar e estimular | Grupo - 1

gestão financeira da TAP | performance. - 1

apoio jurídico do Centro Integrador do Grupo. | Grupo - 1

gestão financeira da TAP | controlar e estimular - 1

gestão financeira da TAP | Unidades de Negócio - 1

Grupo | Planeamento Estratégico do Grupo Missão Principal - 1

Grupo | controlar e estimular - 1

Grupo | Recursos Humanos do Grupo Missão Principal - 1

Administrador-Delegado. | Administrador-Delegado Missão Principal - 1

Presidente | executivo Administrador-Delegado - 1

gestão financeira da TAP | Grupo - 1

controlar e estimular | Unidades de Negócio - 1

aproveitamento da posição consolidada dos recursos financeiros do Grupo | gestão financeira da TAP - 1

gestão financeira da TAP | aproveitamento da posição consolidada dos recursos financeiros do Grupo - 1

performance. | controlar e estimular - 1

Vice-Presidentes Executivos Missão Principal | Presidente - 1

gestão de Recursos Humanos | Grupo - 1

controlar e estimular | gestão financeira da TAP - 1 apoio jurídico do Centro Integrador do Grupo. | Centro Integrador do Grupo - 1

controlar e estimular | aproveitamento da posição consolidada dos recursos financeiros do Grupo - 1

Planeamento Estratégico do Grupo Missão Principal | Grupo Missão Principal - 1

aproveitamento da posição consolidada dos recursos financeiros do Grupo | Unidades de Negócio - 1

Grupo | aproveitamento da posição consolidada dos recursos financeiros do Grupo - 1

poderes de gestão da sociedade | C.A. - 1

performance. | aproveitamento da posição consolidada dos recursos financeiros do Grupo - 1

Grupo | gestão financeira da TAP - 1

executivo Administrador-Delegado | Presidente - 1

Grupo | Assessoria Jurídica do Grupo Missão Principal - 1

gestão de Recursos Humanos. | Grupo - 1

Planeamento Estratégico do Grupo Missão Principal | Grupo - 1

Unidades de Negócio | aproveitamento da posição consolidada dos recursos financeiros do Grupo - 1

performance. | gestão financeira da TAP - 1

aproveitamento da posição consolidada dos recursos financeiros do Grupo | Grupo - 1

Administrador-Delegado Missão Principal | Administrador-Delegado. - 1

Assessoria Jurídica do Grupo Missão Principal | Grupo Missão Principal - 1

performance. | Grupo - 1

Grupo | apoio jurídico do Centro Integrador do Grupo. - 1

performance. | Unidades de Negócio - 1

C.A. | poderes de gestão da sociedade - 1

Grupo Missão Principal | Assessoria Jurídica do Grupo Missão Principal - 1

aproveitamento da posição consolidada dos recursos financeiros do Grupo | performance. - 1

Unidades de Negócio | gestão financeira da TAP - 1

Grupo | performance. - 1

termos da subdelegação de competências do Administrador-Delegado | supervisão - 1

termos da lei e dos estatutos da empresa | assuntos de administração da sociedade - 1

Unidades de Negócio | performance. - 1

controlar e estimular | performance. - 1

Recursos Humanos do Grupo Missão Principal | Grupo - 1

Assessoria Jurídica do Grupo Missão Principal | Grupo - 1

assuntos de administração da sociedade | termos da lei e dos estatutos da empresa - 1

Centro Integrador do Grupo | apoio jurídico do Centro Integrador do Grupo. - 1

Grupo | gestão de Recursos Humanos - 1

supervisão | termos da subdelegação de competências do Administrador-Delegado - 1 aproveitamento da posição consolidada dos recursos financeiros do Grupo | controlar e estimular - 1

Presidente | Vice-Presidentes Executivos Missão Principal - 1

Grupo Missão Principal | Planeamento Estratégico do Grupo Missão Principal - 1

Grupo | gestão de Recursos Humanos. - 1

Unidades de Negócio | Grupo - 1

Unidades de Negócio | controlar e estimular - 1

Grupo Missão Principal | Recursos Humanos do Grupo Missão Principal - 1

Grupo | Unidades de Negócio - 1

Recursos Humanos do Grupo Missão Principal | Grupo Missão Principal - 1

Co-ocorrencia indexada

termos da subdelegação de competências do Administrador-Delegado - supervisão (1)

supervisão - termos da subdelegação de competências do Administrador-Delegado (1)

apoio jurídico do Centro Integrador do Grupo. - Centro Integrador do Grupo (1) Grupo (1)

Centro Integrador do Grupo - Grupo (2) apoio jurídico do Centro Integrador do Grupo. (1)

C.A. - poderes de gestão da sociedade (1)

poderes de gestão da sociedade - C.A. (1)

controlar e estimular - gestão financeira da TAP (1) performance. (1) Grupo (1) Unidades de Negócio (1) aproveitamento da posição consolidada dos recursos financeiros do Grupo (1)

aproveitamento da posição consolidada dos recursos financeiros do Grupo - gestão financeira da TAP (1) performance. (1) controlar e estimular (1) Grupo (1) Unidades de Negócio (1)

Unidades de Negócio - gestão financeira da TAP (1) performance. (1) controlar e estimular (1) Grupo (1) aproveitamento da posição consolidada dos recursos financeiros do Grupo (1)

gestão financeira da TAP - performance. (1) controlar e estimular (1) Grupo (1) Unidades de Negócio (1) aproveitamento da posição consolidada dos recursos financeiros do Grupo (1)

performance. - gestão financeira da TAP (1) controlar e estimular (1) Grupo (1) Unidades de Negócio (1) aproveitamento da posição consolidada dos recursos financeiros do Grupo (1)

Assessoria Jurídica do Grupo Missão Principal - Grupo Missão Principal (1) Grupo (1)

Recursos Humanos do Grupo Missão Principal - Grupo Missão Principal (1) Grupo (1)

Grupo Missão Principal - Grupo (4) Recursos Humanos do Grupo Missão Principal (1) Planeamento Estratégico do Grupo Missão Principal (1) Assessoria Jurídica do Grupo Missão Principal (1)

Planeamento Estratégico do Grupo Missão Principal - Grupo Missão Principal (1) Grupo (1)

Grupo - Grupo Missão Principal (4) Centro Integrador do Grupo (2) gestão financeira da TAP (1) performance. (1) Planeamento Estratégico do Grupo Missão Principal (1) apoio jurídico do

Centro Integrador do Grupo. (1)   gestão de Recursos Humanos (1)   Unidades de Negócio (1)
gestão de Recursos Humanos. (1)   aproveitamento da posição consolidada dos recursos financeiros do Grupo (1)   Assessoria Jurídica do Grupo Missão Principal (1)   Recursos Humanos do Grupo Missão Principal (1)   controlar e estimular (1)

Vice-Presidentes Executivos Missão Principal - Presidente (1)

executivo Administrador-Delegado - Presidente (1)

Presidente - executivo Administrador-Delegado (1)   Vice-Presidentes Executivos Missão Principal (1)

termos da lei e dos estatutos da empresa - assuntos de administração da sociedade (1)

assuntos de administração da sociedade - termos da lei e dos estatutos da empresa (1)

Administrador-Delegado Missão Principal - Administrador-Delegado. (1)

Administrador-Delegado. - Administrador-Delegado Missão Principal (1)

gestão de Recursos Humanos - gestão de Recursos Humanos. (2)   Grupo (1)

gestão de Recursos Humanos. - gestão de Recursos Humanos (2)   Grupo (1)

A tabela 2 faz-nos uma listagem simplificada das co-ocorrências entre termos listadas acima, apresentando os resultados em termos de índices, tanto das frases como das Unidades Terminológicas.

Frase (Index)	UTs
0	8, 26
1	6,18, 31
2	30
3	2, 6,23
4	10, 21
5	17, 22
6	3, 19
7	8, 28
8	11
9	20
10	6, 17, 22
11	11, 15
12	2, 6
13	6, 18
14	14
15	6, 18, 27
16	6,13,18
17	4, 5, 6, 7, 24, 25
18	1, 12

Tabela 2 – Tabela dos termos por frases

Da tabela 2 pode-se constatar uma serie de co-ocorrências entre termos, tais como 6 e 18 que co-ocorrem nas frases 1, 13, 15 e 16.

3.8 Análise dos resultados

Relativamente á análise dos resultados obtidos o que se pretende é obter um feedback sobre o comportamento do sistema, ou seja, fazer um controlo de qualidade, submetendo o sistema a um teste de qualidade recorrendo aos critérios de avaliação definidos na secção 3.6.

Avaliação

Retornando ao exemplo 3.7.1, e estando agora em condições de avaliar, podemos fazer o seguinte levantamento de dados:

Total de Informação Recuperada (N) = 10,

Consideremos duas amostras de 5 elementos cada:
Para a primeira amostra (os 5 primeiros elementos) temos uma Informação Relevante (n) = 4

Com isso

$$P = \frac{4}{5} \times 100\% = 80\%$$

$$R = \frac{4}{8} \times 100\% = 50\%$$

$$F = \frac{2 \times 0.5 \times 0,8}{0.5 + 0,8} \times 100\% = 61,537\%$$

Para a segunda amostra temos

n = 6

$$P = \frac{2}{5} \times 100\% = 40\%$$

$$R = \frac{6}{8} \times 100\% = 75\%$$

$$F = \frac{2 \times 0.75 \times 0,4}{0.75 + 0,4} \times 100\% = 52,17\%$$

A tabela 3 mostra estes resultados.

n	N	P(%)	R(%)	F(%)
4	5	80	50	61,5
6	10	40	75	52,2

Tabela 3 – valores experimentais de Precisão de Recall

O exemplo acima constitui uma pequena demonstração da extracção de termos dum parágrafo, como dito anteriormente. Contudo, a tabela 4 mostra-nos o resultado da extracção sobre o nosso corpo de texto de teste com cerca de 3481 tokens.

A extracção manual encontra-se em anexo (Anexo A.3). Dos 3481 tokens de entrada, conseguiu-se extrair cerca de 189 UTs manuais considerados válidos. Feito isto, o passo seguinte seria fazer um cruzamento de dados entre os resultados manuais e automáticos onde se chegou ao conjunto de resultados apresentados na tabela 2, considerados satisfatórios.

Sabendo que o sistema conseguiu recuperar 360 UTs, o passo seguinte é fazer pequenos lotes de amostra, que neste caso é 1/10 do total de UTs recuperados (função compare() – Anexo A.4)

Candidatos(n)	N	P(%)	R(%)	F(%)
21	36	58,33	11,11	18,67
27	72	37,50	14,29	20,69
33	108	30,56	17,46	22,22
39	144	27,08	20,63	23,42
44	180	24,44	23,28	23,85
54	216	25,00	28,57	26,67
59	252	23,41	31,22	26,76
67	288	23,26	35,45	28,09
74	324	22,84	39,15	28,85
78	360	21,67	41,27	28,42
Total manual	**189**			
Total sistema	**360**			

Tabela 4 – Valores experimentais de Precisão, Recall e medida F

De acordo com a tabela 4 no primeiro lote temos uma Precisão de 58,33 %, e este valor vai diminuindo consoante se aumenta o número de termos. Isto mostra que há uma forte tendência de encontrar termos relevantes nos primeiros grupos, termos estes que ocorrem com maior frequência.

À medida que se acrescenta o número de termos, aumenta a sensibilidade (R), ou seja, para grupos maiores de termos, maior é o número de termos válidos. Contudo a Precisão e o Recall são inversamente proporcionais, ou seja, o que se ganha com a Precisão perde-se com a sensibilidade, como mostra o gráfico da figura 7.

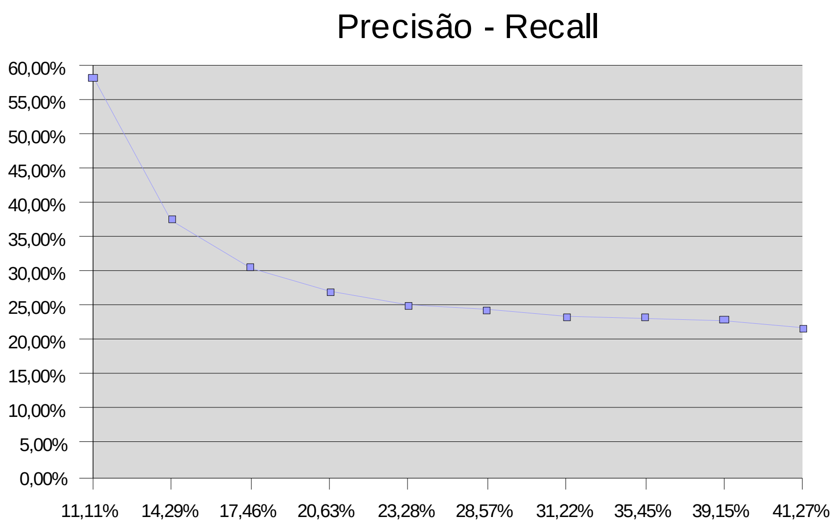

Fig. 7 - Gráfico da Precisão em função do Recall

De acordo com o gráfico da figura 7, à medida que aumentamos o número de amostras, o sistema torna-se menos preciso e mais sensível. Podemos concluir também que a precisão tende para um valor constante, sobretudo para um número elevado de amostras. Isto deve-se ao facto de o sistema colocar no topo da lista os termos que ocorrem com maior frequência. Sendo assim, quando retiramos um lote com X primeiros termos desta lista, há uma grande probabilidade destes serem termos candidatos, fazendo com que a precisão seja alta. Ao contrário disto o sistema vai-se tornando cada vez mais sensível uma vez que o número de termos candidatos vai-se aumentando.

Uma forma de monitorizar o comportamento do sistema é analisar a medida F, que como definido anteriormente, é a relação entre a Precisão de o Recall.

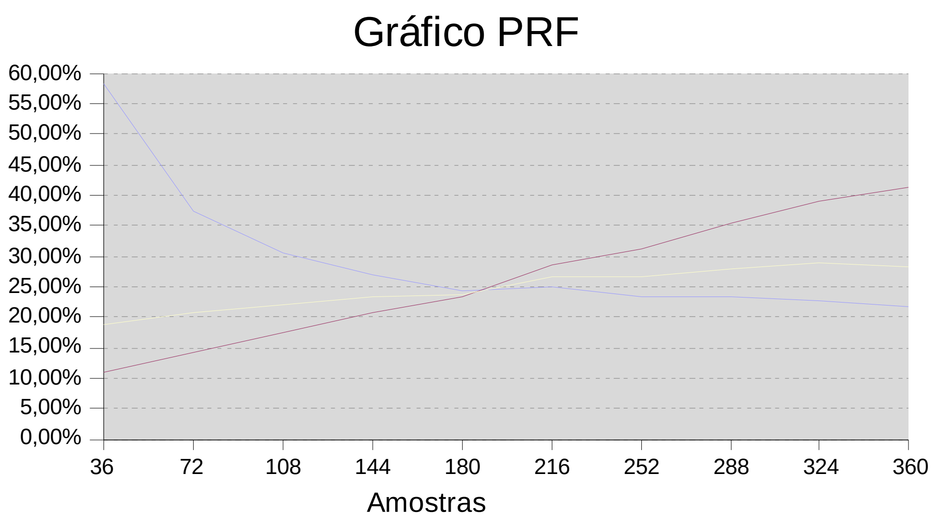

Fig. 8 - Gráfico da Precisão, Recall e Medida F em função do número de amostras (360 UTs)

Na figura 8 a Precisão está representada a Azul, Recall a violeta, e medida F a rosa. A curva da função F é dada por uma parábola (concavidade voltada para baixo) onde atinge um máximo (ponto óptimo), que neste caso é de 28,85%. Este ponto é atingido quando a Precisão e o Recall tendem a ficar constantes, e a partir dessa zona o F começa a diminui.
O máximo de F constitui um ponto de viragem na análise do sistema, sendo a zona a jusante uma zona menos apropriada, uma vez que a Precisão e o Recall não muito alteram.

Voltando ao exemplo da secção 3.7.1 e tendo em vista os termos anteriormente extraídos, tanto de conforma automática, como manualmente, a avaliação consiste numa comparação entre as duas listas. Ou seja, ver o quão eficaz é o sistema de extracção automática é em relação ao que realmente se pretende.

A tabela 5 apresenta os resultados do exemplo 3.7.1

n	N	P(%)	R(%)	F(%)
1	1	100	14,28571	25
1	2	50	14,28571	22,22222
1	3	33,33333	14,28571	20
1	4	25	14,28571	18,18182
2	5	40	28,57143	33,33333
3	6	50	42,85714	46,15385
3	7	42,85714	42,85714	42,85714
3	8	37,5	42,85714	40
4	9	44,44444	57,14286	50
4	10	40	57,14286	47,05882
Total manual	**7**			
Total sistema	**10**			

Tabela 5 – tabela da Precisão, Recall e Medida F

A figura 9 apresenta o traçado gráfico da tabela 5.

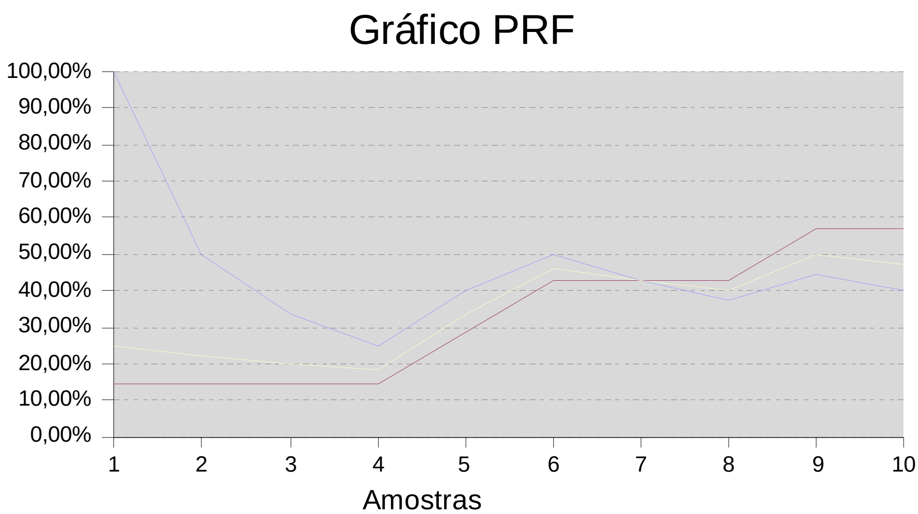

Fig. 9 - Gráfico da Precisão, Recall e Medida F em função do número de amostras (10 UTs)

Ao compararmos o gráfico da figura 9 com o gráfico da figura 8, notamos alguma diferença a nível comportamental, querendo dizer com isto que o gráfico da figura 8 mostra-se mais estável, apresentando menos oscilações e variações bruscas do que o gráfico da figura 9. Isto deve-se ao facto do sistema representado pelo gráfico da figura 8 ter um maior número de amostras, ou seja, um sistema maior, demonstrando com isto que este método de análise é mais fiável quanto maior for o sistema em estudo.

Capítulo 4

Conclusão e Trabalho Futuro

Neste projecto apresentou-se um sistema capaz de fazer uma extracção semântica a partir de um ficheiro de texto, baseado em filtro lexical para eliminar termos indesejados, tendo em conta o modelo de dados que representa o conjunto de conceitos previamente definido, de acordo com o domínio em questão. Vimos anteriormente que a pesquisa semântica é uma tecnologia relativamente recente, constituindo o pilar da Web 3.0.

O Portal DOV contém um motor de pesquisa desenvolvido num projecto anterior [1] que é uma estrutura de suporte e coordenação on-line às actividades desenvolvidas pelos tripulantes e os utilizadores dos serviços abrangidos pelo Portal.

Sabendo que o Portal faz pesquisas sobre documentos tais como manuais de voos, Fichas de discursos, etc, apresentando os dados sob a forma de uma tabela com os campos de Relevância, Nome do documento, Data de emissão e Serviço, e embora os resultados alcançados são considerados satisfatórios, isto mesmo assim fica a quem das expectativas de responder à questão posta na parte introdutória deste projecto, capítulo 1: "qual é o limite máximo de bagagem de mão?"

Para tal o sistema teria que ser capaz de fazer pesquisas não só sobre os títulos, mas também ser capaz pesquisar conteúdos destes documentos.

Por outras palavras, é dizer que o sistema teria que recuperar informações intrínsecas, ou seja, informações contidas dentro dos documentos que são sujeitos a análises pelo motor. Por exemplo, a ocorrência de um determinado termo num documento e sua localização no texto são informações levadas em consideração durante a elaboração do ranking dos resultados, onde a ordem de listagem é no sentido decrescente de relevância.

Uma vez que este protótipo faz extracção de informações intrínsecas, apresentando os dados sob a forma de uma lista indexada de Termos Semânticos, no sentido decrescente de relevância, o futuro passa pela fusão destes dois sistemas, integrando este protótipo no Portal DOV, aumentando a sua capacidade de pesquisa.

Um projecto futuro será o desenvolvimento dum módulo capaz de interligar estes dois sistemas, fazendo com que a lista de extracção semântica seja objecto de pesquisa do motor.

Referências

[1] Luís Ângelo de Sá Barbosa, “Motor de Pesquisa sobre base de dados documental do Portal DOV”. Relatório de Estágio Curricular do MIEIC 2006/2007, FEUP, Porto.

[2] Tim Berners-Lee, James Hendler e Ora Lassila,” Web Semântica: um novo formato de conteúdo para a Web que tem significado para computadores vai iniciar uma revolução de novas possibilidades.”

[3] Tropes, Semantic Analysis - Quick Guide http://www.semantic-knowledge.com/doc/V70/text-analysis/quick-guide.htm

[4] Powerset, Intelligent Semantic Search Engine. http://www.powerset.com/

[5] Netcraft, Market Share for Top Servers Across All Domains August 1995 - May 2009. http://news.netcraft.com/

[6] Sarmento, Luís and Oliveira, Eugénio.: 2007 “Making RAPOSA (FOX) smarter”, In Alessandro Nandi and Carol Peters (Eds.) Working Notes for the CLEF 2007 workshop, 19-21 September, Budapest, Hungary.

[7] ISO 1087, 1 ª edição, 1990, Terminology – Vocabulary http://www.iso.org/iso/catalogue_detail.htm?csnumber=5591

[8] Thurmair, Gr., 2000: TQPro: Quality Tools for the Translation Process. Proc. ASLIB 2000, London.

[9] ISO 9126, Software engineering -- Product quality. http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=22749

[10] System QUIRK: System QUIRK, Language Engineering Workbench. http://www.mcs.surrey.ac.uk/SystemQ/

[11] R. Eletr. de Com. Inf. Inov. Saúde. Rio de Janeiro, v. 1, n. 1, p. 117-121, jan-jun, 2007

[12] Arppe, A.; "Term Extraction from Unrestricted Text. Proc." NODALIDA'97; Helsinki,1995.

[13] Bourigault, D.; "A Terminology Extraction Software for Knowledge Acquisition from Texts. In Proceedings of the 9th KAW";LEXTER; Banff, Canada,1995.

[14] Frantzi, K., Ananiadou, S., Mima, H., 2000 Automatic recognition of multi-word terms: the C-value / NC- value method. Int. J. Digit. Lib. 3, 2000: 115-130

[15] Dias, G., et al., 2000: Combining Linguistics with Statistics for Multiword Term Extraction: A Fruitful Association? Proc. RIAO 2000

[16] OLIF, 2002: Open Lexicon Interchange Format http://www.olif.net/

[17] Pantel, P., Lin, D., 2001 A statistical corpus-based term extractor. http://www.cs.ualberta.ca/~lindek/papers/ai01.pdf

[18] Manual de Regulamentação de Operações de Voo, TAP Air Portugal, Maio 2007.

[19] Heidemann, B., Volk, M., 1999: Evaluation of Terminology Extraction Tools. Report Univ. Zürich.

[20] Sauron, V., 2002: Tearing out the Terms: Evaluating Terms Extractors. Proc. ASLIB 2002, London.

Anexo A

A.1 – tokenize()

```
sub Tokenize {
  $_ = $_[0];
  s/\s+/\n/g;                       ## put every word on a different line
  s/^\n//;                          ## remove possible initial empty line
  s/([.,!?:;,])\n/\n$1\n/g;         ## put punctuation signs on separate line
  s/\n(["'`])([^\n])/\n$1\n$2/g;    ## put initial quote on separate line
  s/([^\n])(["'`])\n/$1\n$2\n/g;    ## put final quote on separate line
  s/([^\n])([.,])\n/$1\n$2\n/g;     ## put punctuation before " on separate line
  s/\n([A-Z])\n\./\n$1./g;          ## take care of initials in names
  s/\n\.\n([^"A-Z])/\.\n$1/g;       ## put periods behind abbreviations
  s/(\.[A-Z]+)\n\.\n/$1.\n/g;       ## put periods behind abbreviations
  s/([^\n])'s\n/$1\n's\n/g;         ## move final 's to separate line
  s/([^\n])n't\n/$1\nn't\n/g;       ## move final n't to separate line
  s/([^\n])'re\n/$1\n're\n/g;       ## move final 're to separate line
  s/\n\$([^\n])/\n\$\n$1/g;         ## move initial $ to separate line
  s/([^\n])%\n/$1\n%\n/g;           ## move final % to separate line
  return(split(/\n/,$_));
}
```

A.2 - n-gram.pl

```
## Projecto:  PPSAMBD FEUP 2007/08;
## Autor: Luís Barreto,luisbarreto25@gmail.com;
## !c:\Perl\bin

#require "./parameters.pl";
use strict;
##use warnings;
use FileHandle;

sub Tokenize {
  $_ = $_[0];
  s/\s+/\n/g;                        ## put every word on a different line
  s/^\n//;                           ## remove possible initial empty line
  s/([.,!?:;,])\n/\n$1\n/g;          ## put punctuation signs on separate line
  s/\n(["'`])([^\n])/\n$1\n$2/g;     ## put initial quote on separate line
  s/([^\n])(["'`])\n/$1\n$2\n/g;     ## put final quote on separate line
  s/([^\n])([.,])\n/$1\n$2\n/g;      ## put punctuation before " on separate line
  s/\n([A-Z])\n\./\n$1./g;           ## take care of initials in names
  s/\n\.\n([^"A-Z])/\.\n$1/g;        ## put periods behind abbreviations
  s/(\.[A-Z]+)\n\.\n/$1.\n/g;        ## put periods behind abbreviations
  s/([^\n])'s\n/$1\n's\n/g;          ## move final 's to separate line
  s/([^\n])n't\n/$1\nn't\n/g;        ## move final n't to separate line
  s/([^\n])'re\n/$1\n're\n/g;        ## move final 're to separate line
  s/\n\$([^\n])/\n\$\n$1/g;          ## move initial $ to separate line
  s/([^\n])%\n/$1\n%\n/g;            ## move final % to separate line
  return(split(/\n/,$_));
}

## open and read input file
sub ProcessFileNgram($$$) {
  my $filename = shift; ## fi_in
  my $n = shift;
  my $filename_out = shift;
  my $file_handle = new FileHandle();
  if (!$file_handle->open($filename)) {
    die("Could not open input file '$filename'\n");
```

```
  }
  my %result_hash = ();
  while (!$file_handle->eof()) {
    my $line = $file_handle->getline();
    my @tokens = Tokenize($line);
    Ngram(\@tokens,$n, \%result_hash);
  }
  $file_handle->close();

  ## Let's output the final result fo file
  PrintFile($filename_out,\%result_hash);

}

sub Ngram($$$) {
  my $token_ref = shift;
  my $n = shift;
  my $hash_ref = shift;
  my @tokens = @{$token_ref};
  for(my $ii = 0; $ii <= $#tokens - ($n - 1); $ii++) {
    my $item = "";
    for(my $jj = 0; $jj <= $n - 1; $jj++) {
      $item .= $tokens[$ii + $jj] . " ";
    }
    if($item =~ /[a-zA-Zàé]/) {
      $$hash_ref{$item}++;
    } ## if
  } ## for
}

## open and write output file
sub PrintFile($$) {
  my $filename = shift;
  my $hash_ref = shift;
  my %hash = %{$hash_ref};
  my $file_handle = new FileHandle();
  if (!$file_handle->open(">$filename")) {
    die("Could not create output file '$filename'\n");
```

```
  } ## if
  ## put hash keys in list and sort the list
  my @keys = sort({$hash{$b} <=> $hash{$a}} keys %hash);
  foreach my $key (@keys) {
    print $file_handle "$key $hash{$key}\n";
  } ## foreach
  $file_handle->close();
}
```

A.3 – UTs manualmente extraídas

MACROESTRUTURA TAP

REGULAMENTAÇÃO LABORAL

MANUAL DE REGULAMENTAÇÃO DE OPERAÇÕES DE VOO

ACORDO DE EMPRESA - PNT

ACORDO DE EMPRESA - PNC

REGULAMENTAÇÃO INTERNA

Cláusula 49ª

cláusula

cadeiras de classe

classe executiva

passageiros

aterragem

território de Portugal continental

Tempos de serviço de voo

tempos de serviço de voo

horas de voo

longo-curso

pilotos

voo

número de aterragens

cláusulas 50ª

Tripulações reforçadas
tripulações reforçadas
tripulação
certificação
avião
meios de descanso
tempos de voos
comandos
descanso
bordo
tripulação reforçada
reforço completo
reforço
SPAC
rede TAP
Cláusula 51ª
situação de extracrew
número de aterragens
TSV
tripulante
SISTEMA DE PRETENSÕES INDIVIDUAIS
Cláusula 52ª
Pretensões individuais
pretensões individuais
escala pessoal
pretensão de fixar
voos
operação
custos adicionais
escala mensal
pretensão
valor total de pontos de crédito
Transporte Aéreo
número de irregularidades
controlo da operação
irregularidade
Unidade de Negócio
profissionais

apoio administrativo e de logística
áreas de Transporte Aéreo
funções contabilísticas e de apuramento de resultados
Unidade de Negócio
apuramento e processamento das contas do Transporte Aéreo
sistemas de informação da Unidade de Negócio
OPERAÇÕES DE VOO DIRECÇÃO GERAL
gestão técnica de tripulações (Portugal e Exterior)
formação de tripulações (PNT e PNC)
planeamento de escalas de Pessoal Navegante
Airline Operations Policy Manual AOPM
serviços de voo
segurança de voo
Operações de voo
base de dados
Serviços de Segurança de Voo (operadores e construtores)
ACORDO DE EMPRESA – PNC
Cláusula 3.a
Categorias profissionais
profissão de tripulante de cabina
entidade aeronáutica nacional
Supervisor de cabina (S/C)
equipamentos de wide body
equipamentos de wide body e narrow body
serviço de cabina
cumprimento das normas de segurança
conforto e segurança durante o voo
verificação dos itens de segurança
exercício das funções de S/C
cumprimento dos procedimentos de segurança
avião
ocupantes
chefes de zona (C/Cs)
C/Cs
carga do avião
alimentação dos passageiros e tripulantes
rotinas do serviço de cabina
Chefe de cabina (C/C)

serviço de zona
cabina
funções do C/C
Lugares de descanso e tomada de refeição
voos de longo curso
aviões de WB
Crew Rest Seats Model SH
cortinas semi rígidas
elementos da tripulação
tripulantes da cabina
TAP
fabricante do avião
SNPVAC
Escalas de serviço
escalas de serviço
períodos de serviço de voo
hora de apresentação e duração de serviço dos períodos de voo
duty e duty pay
blockpay
horas dos serviços de assistência
serviço on call
serviços de reserva de vinte e quatro horas
trabalho no solo
escalas de serviço mensais
períodos de serviço de voo
serviço on call
serviço de assistência e de serviço de reserva de vinte e quatro horas
Gabinete de acompanhamento
planeamento das folgas semanais
grau de fiabilidade
planeamento das operações de voo
necessidades do transporte aéreo
pessoal navegante
Cooperação TAP/SNPVAC
distribuição dos serviços de voo
Planeamento mensal dos tripulantes
Registo da actividade realizada pelos tripulantes
registo das irregularidades

planeamentos dos tripulantes
serviço assistência
período de serviço de voo
início da folga semanal
planeamento de voos subsequente do tripulante
SUBSÍDIOS
Escala de Abono do Subsídio
subsídios
escalas
hora de apresentação
atrasos à saída dos voos
escala tap refeição
folha de vencimentos do tripulante
aplicação da presente regulamentação
Valor do Subsídio "On Ground"
valor do subsídio
CTE PORTUGAL CONTINENTAL REGIÕES AUTÓNOMAS EUROPA ÁFRICA OUTRAS O/P
ponto corresponde ao valor monetário de PTE 2$83 (DOIS ESCUDOS E OITENTA E TRÊS CENTAVOS PORTUGUESES)
vencimento do tripulante
Subsídio de Pernoita
subsídio de pernoita
ATA
Abono do Subsídio
recebimento do subsídio de pernoita
prolongamento de uma estadia
subsídio eventual
Estadia fora da base
escala de estacionamento
extravio temporário
bagagem de porão de tripulante em serviço de voo
pagamento do subsídio
montante do subsídio eventual
tripulante em serviço de voo
bagagem extraviada ao dispor do hotel
pagamento de subsídios
extravio temporário de bagagem de porão
folha de subsídios

APARÊNCIA
Aparência
serviço da empresa
PN
manter o uniforme limpo e apresentável
artigos de "toilette" e vestuário extra
uso de essências fortes
uso de óculos de correcção de lentes incolores ou fotocromáticas claras
uso de óculos de lentes espelhadas ou coloridas
nterior da cabina de passageiros do avião
armações dos óculos
condição física do PN
corte do cabelo e o tipo de penteado
base da gola do casaco e lateral a base da orelha
cabelo deve manter uma cor natural e uniforme
uso de gel de aspecto molhado
uso de barba e bigode
barba
bigode
período de transição de crescimento de barba ou bigode
corte de barba tipo
mal aparado

A.4 – Avaliation.pl

```
## Projecto:  PPSAMBD FEUP 2007/08;
## Autor: Luís Barreto,luisbarreto25@gmail.com;
## !c:\Perl\bin
use strict;
##use warnings;
use FileHandle;
```

```
my %manual = "";



sub ManualTerms() {
  my $file = "expressoes.txt";
  chomp($file);
  my $file_handle = new FileHandle();
  if (!$file_handle->open($file)) {
    die("Could not open input file '$file'\n");
  }
  my @tokens = ();
  while (!$file_handle->eof()) {
    my $line = $file_handle->getline();
    $line =~ s/^\s+//;  ##remove possible inicial white space
    $line =~ s/^\s+$//; ##remove possible final white space
    $line =~ s/^\n*//;  ##remove possible inicial white line
    $line =~ s/\n*$//;  ##remove possible final white line
    if($line ne "") {   ## do not accept empty line
       $manual{$line}++;
    }
  }
  $file_handle->close();
}

sub SystemTerms() {
  my $file = "terms.txt";
  chomp($file);
  my $file_handle = new FileHandle();
  if (!$file_handle->open($file)) {
    die("Could not open input file '$file'\n");
  }
  my $line = "";
  my @system = ();
  while (!$file_handle->eof()) {
    $line = $file_handle->getline();
    $line =~ s/^\s+//;  ##remove possible inicial white space
    $line =~ s/^\s+$//; ##remove possible final white space
```

```
    $line =~ s/^\n*//;  ##remove possible inicial white line
    $line =~ s/\n*$//;  ##remove possible final white line
    if($line ne "") {   ## do not accept empty line
       push(@system,$line);
    }
  }
  #print @system;
  Compare(\@system);
  $file_handle->close();
}

sub Compare($) {
       my $system_ref = shift;
       my @system = @{$system_ref};
  my $filename_out = "result.txt";
  ## put hash keys in list and sort the list
  my @manual = sort({$manual{$b} <=> $manual{$a}} keys %manual);
  ##print @manual;
  my $file_handle = new FileHandle();
  if (!$file_handle->open(">$filename_out")) {
    die("Could not open input file '$filename_out'\n");
  }
       my %truelist = "";
       my $total = 0;
  my $total_manual = $#manual;
  my $total_system = $#system;
  my
($count1,$count2,$count3,$count4,$count5,$count6,$count7,$count8,$count9,$coun
t10,$count11,$count12) = 0;
  my @truelist = ();
  for(my $ii = 0; $ii <= $total_manual; $ii++) {
    for(my $jj = 0; $jj <= $total_system*1/10; $jj++) {
       if($manual[$ii] eq $system[$jj]) {
             $count1 ++;
       }
       } ## for $jj
       for(my $jj = 0; $jj <= $total_system*2/10; $jj++) {
       if($manual[$ii] eq $system[$jj]) {
```

```
        $count2 ++;
}
} ## for $jj
for(my $jj = 0; $jj <= $total_system*3/10; $jj++) {
if($manual[$ii] eq $system[$jj]) {
        $count3 ++;
}
} ## for $jj
for(my $jj = 0; $jj <= $total_system*4/10; $jj++) {
if($manual[$ii] eq $system[$jj]) {
        $count4 ++;
}
} ## for $jj
for(my $jj = 0; $jj <= $total_system*5/10; $jj++) {
if($manual[$ii] eq $system[$jj]) {
        $count5 ++;
}
} ## for $jj
for(my $jj = 0; $jj <= $total_system*6/10; $jj++) {
if($manual[$ii] eq $system[$jj]) {
        $count6 ++;
}
} ## for $jj
for(my $jj = 0; $jj <= $total_system*7/10; $jj++) {
if($manual[$ii] eq $system[$jj]) {
        $count7 ++;
}
} ## for $jj
for(my $jj = 0; $jj <= $total_system*8/10; $jj++) {
if($manual[$ii] eq $system[$jj]) {
        $count8 ++;
}
} ## for $jj
for(my $jj = 0; $jj <= $total_system*9/10; $jj++) {
if($manual[$ii] eq $system[$jj]) {
        $count9 ++;
}
} ## for $jj
```

```
        for(my $jj = 0; $jj <= $total_system; $jj++) {
        if($manual[$ii] eq $system[$jj]) {
              $count10 ++;
              push(@truelist,$system[$jj]);
        }
        } ## for $jj
    } ## for $ii

    my
($p1,$p2,$p3,$p4,$p5,$p6,$p7,$p8,$p9,$p10,$r1,$r2,$r3,$r4,$r5,$r6,$r7,$r8,$r9,
$r10) = 0;
    my ($f1,$f2,$f3, $f4,$f5, $f6,$f7,$f8,$f9,$f10) = 0;
    $p1 = $count1/($total_system*1/10) * 100;
    $p2 = $count2/($total_system*2/10) * 100;
    $p3 = $count3/($total_system*3/10) * 100;
    $p4 = $count4/($total_system*4/10) * 100;
    $p5 = $count5/($total_system*5/10) * 100;
    $p6 = $count6/($total_system*6/10) * 100;
    $p7 = $count7/($total_system*7/10) * 100;
    $p8 = $count8/($total_system*8/10) * 100;
    $p9 = $count9/($total_system*9/10) * 100;
    $p10 = $count10/($total_system) * 100;

    $r1 = $count1/$total_manual * 100;
    $r2 = $count2/$total_manual * 100;
    $r3 = $count3/$total_manual * 100;
    $r4 = $count4/$total_manual * 100;
    $r5 = $count5/$total_manual * 100;
    $r6 = $count6/$total_manual * 100;
    $r7 = $count7/$total_manual * 100;
    $r8 = $count8/$total_manual * 100;
    $r9 = $count9/$total_manual * 100;
    $r10 = $count10/$total_manual * 100;

    $f1 = $p1*$r1*2/($p1+$r1);
    $f2 = $p2*$r2*2/($p2+$r2);
    $f3 = $p3*$r3*2/($p3+$r3);
    $f4 = $p4*$r4*2/($p4+$r4);
```

```
  $f5 = $p5*$r5*2/($p5+$r5);
  $f6 = $p6*$r6*2/($p6+$r6);
  $f7 = $p7*$r7*2/($p7+$r7);
  $f8 = $p8*$r8*2/($p8+$r8);
  $f9 = $p9*$r9*2/($p9+$r9);
  $f10 = $p10*$r10*2/($p10+$r10);

  print $file_handle "\nManual  = $total_manual\nSistema =
$total_system\n\n\n";
  print $file_handle "n = $count1 N = ".int($total_system*1/10)."  P = $p1%  R
= $r1%  F= $f1%\n";
  print $file_handle "n = $count2 N = ".int($total_system*2/10)."  P = $p2%  R
= $r2%  F= $f2%\n";
  print $file_handle "n = $count3 N = ".int($total_system*3/10)." P = $p3%  R
= $r3%  F= $f3%\n";
  print $file_handle "n = $count4 N = ".int($total_system*4/10)." P = $p4%  R
= $r4%  F= $f4%\n";
  print $file_handle "n = $count5 N = ".int($total_system*5/10)." P = $p5%  R
= $r5%  F= $f5%\n";
  print $file_handle "n = $count6 N = ".int($total_system*6/10)." P = $p6%  R
= $r6%  F= $f6%\n";
  print $file_handle "n = $count7 N = ".int($total_system*7/10)." P = $p7%  R
= $r7%  F= $f7%\n";
  print $file_handle "n = $count8 N = ".int($total_system*8/10)." P = $p8%  R
= $r8%  F= $f8%\n";
  print $file_handle "n = $count9 N = ".int($total_system*9/10)." P = $p9%  R
= $r9%  F= $f9%\n";
  print $file_handle "n = $count10 N = ".int($total_system*10/10)." P = $p10%
R = $r10%  F= $f10%\n";
#  #print $total_system;
  $file_handle->close();
  Truelist (\@truelist);
}

sub Truelist($) {
       my $truelist_ref = shift;
       my @truelist = @{$truelist_ref};
       #print @truelist;
```

```
        my $filename_out = "truelist.txt";
        chomp($filename_out);
    my $file_handle = new FileHandle();
    if (!$file_handle->open(">$filename_out")) {
      die("Could not open input file '$filename_out'\n");
    }
    for(my $ii = 0; $ii <= $#truelist; $ii++) {
        print $file_handle "$truelist[$ii]\n";
    }

    $file_handle->close();
}

ManualTerms();
SystemTerms();


exit(0);
```

A.5 – Dictionary.pl

```
## Projecto:  PPSAMBD FEUP 2007/08;
## Autor: Luís Barreto,luisbarreto25@gmail.com;
## !c:\Perl\bin
use strict;
##use warnings;
use FileHandle;
my %start = ();
my %stop = ();
my %forbidden = ();

sub Tokenize {
  $_ = $_[0];
```

```
  s/\s+/\n/g;                     ## put every word on a different line
  s/^\n//;                        ## remove possible initial empty line
  s/\n*$//;                       ##remove possible final white line
  s/[()]//g;
  s/([.,!?:;,ª])\n/\n$1\n/g;      ## put punctuation signs on separate line
  s/\n(["'`])([^\n])/\n$1\n$2/g;  ## put initial quote on separate line
  s/([^\n])(["'`])\n/$1\n$2\n/g;  ## put final quote on separate line
  s/([^\n])([.,])\n/$1\n$2\n/g;   ## put punctuation before " on separate line
  s/\n([A-Z])\n\./\n$1./g;        ## take care of initials in names
  s/\n\.\n([^"A-Z])/\.\n$1/g;     ## put periods behind abbreviations
  s/(\.[A-Z]+)\n\.\n/$1.\n/g;     ## put periods behind abbreviations
  s/\n\$([^\n])/\n\$\n$1/g;       ## move initial $ to separate line
  s/([^\n])%\n/$1\n%\n/g;         ## move final % to separate line
  return(split(/\n/,$_));
}

sub ProcessStart() {
  my $start = "start.txt";
  chomp($start);
  my $file_handle = new FileHandle();
  if (!$file_handle->open($start)) {
    die("Could not open input file '$start'\n");
  }
  while (!$file_handle->eof()) {
    my $line = $file_handle->getline();
    my @tokens = split(/ /, $line);
    #print "@tokens";
    my $max = $#tokens;
    for(my $ii = 0; $ii <= $max; $ii++) {
       $start{$tokens[$ii]} = 1;
    }
  }
  $file_handle->close();
}


sub ProcessStop() {
  my $stop = "stop.txt";
```

```
  chomp($stop);
  my $file_handle = new FileHandle();
  if (!$file_handle->open($stop)) {
    die("Could not open input file '$stop'\n");
  }
  while (!$file_handle->eof()) {
    my $line = $file_handle->getline();
    my @tokens = split(/ /, $line);
    my $max = $#tokens;
    for(my $ii = 0; $ii <= $max; $ii++) {
      $stop{$tokens[$ii]} = 1;
    }
  }
  $file_handle->close();
}

sub ProcessForbidden() {
  my $forbidden = "forbidden.txt";
  chomp($forbidden);
  my $file_handle = new FileHandle();
  if (!$file_handle->open($forbidden)) {
    die("Could not open input file '$forbidden'\n");
  }
  while (!$file_handle->eof()) {
    my $line = $file_handle->getline();
    my @tokens = split(/ /, $line);
    my $max = $#tokens;
    for(my $ii = 0; $ii <= $max; $ii++) {
      $forbidden{$tokens[$ii]} = 1;
    }
  }
  $file_handle->close();
}

ProcessStart();
ProcessStop();
ProcessForbidden();
```

```
## open and read input file
sub ProcessFileDict($$) {
  my $filename_in = shift;
  my $filename_out = shift;
  my $file_handle = new FileHandle();
  if (!$file_handle->open($filename_in)) {
    die("Could not open input file '$filename_in'\n");
  }
  my %result_hash = ();
  while (!$file_handle->eof()) {
       my $line = $file_handle->getline();
       #put "§§§bengin§§§" in the begin of the line
       my @tokens = ("§_begin_§");
       #adding to @tokens the returned lines from Tokenize()
    push(@tokens ,Tokenize($line));
    #put "end_of_line" in the end of each line
    push(@tokens, "§_end_§");
    CompareString(\@tokens,\%result_hash);
  }
  $file_handle->close();
  PrintFile($filename_out,\%result_hash);
}

## the compareString function will compare the @c_list elements
## with the @tokens elements.
## for each @tokens,


sub CompareString($$) {
  my $token_ref = shift;
  my $result_hash = shift;
  my @tokens = @{$token_ref};
  my $max = $#tokens - 1;
  for(my $ii = 0; $ii <= $max; $ii++) {
    if($start{lc($tokens[$ii])} == 1) {
      my $delta = extrai($ii + 1, \@tokens, $result_hash);
      $ii += $delta;
    }
```

```
  } ## for
}

sub extrai($$$) {
  my $n = shift;
  my $tokens_ref = shift;
  my $result_hash = shift;
  my @tokens = @{$tokens_ref};
  my $item = "";
  my $max = $#tokens;
  for(my $jj = $n; $jj <= $max; $jj++ ) {
       ## Case 1 - A stop token found
    if($stop{lc($tokens[$jj])} == 1) {
       ## case 2 - A start token was given
       if($start{lc($tokens[$jj - 1])} == 1) {
        return $jj - $n;
       } else {
              $$result_hash{$item}++;
        return $jj - $n;
       }
    }
    ## Case 3 - A forbidden token found
    if($forbidden{lc($tokens[$jj])} == 1) {
       return $jj - $n;
    }
    ## Case 4 - End of line found
    if($jj == $max) {
      return $max - $n;
    }
    $item .= " " . $tokens[$jj];
    $item =~ s/^\s+//;  ##remove possible inicial white space
    $item =~ s/^\s+$//; ##remove possible final white space
    $item =~ s/^\n*//;  ##remove possible inicial white line
    $item =~ s/\n*$//;  ##remove possible final white line
  }
}

## open and write output file
```

```
sub PrintFile($$) {
  my $filename_out = shift;
  my $hash_ref = shift;
  my %hash = %{$hash_ref};
  my $file_handle = new FileHandle();
  if (!$file_handle->open(">$filename_out")) {
    die("Could not open input file '$filename_out'\n");
  }
       my $total = 0;
  ## put hash keys in list and sort the list
  my @keys = sort({$hash{$b} <=> $hash{$a}} keys %hash);
  foreach my $key (@keys) {
    print $file_handle "$key - $hash{$key}\n";
  }

  $file_handle->close();
}
```

A.6 – Co-ocorrency.pl

```
## Projecto:  PPSAMBD FEUP 2007/08;
## Autor: Luís Barreto,luisbarreto25@gmail.com;
## !c:\Perl\bin
use strict;
##use warnings;
use FileHandle;
use Data::Dumper;
my %gbl_term_index = ();
my %gbl_index_terms = ();
my %pair_list = ();
my %co_ocorr_index = ();
my @gbl_terms = ();
my @gbl_phrase = ();
```

```
sub Terms() {
  my $file = "terms.txt";
  chomp($file);
  my $file_handle = new FileHandle();
  if (!$file_handle->open($file)) {
    die("Could not open input file '$file'\n");
  }
  while (!$file_handle->eof()) {
    my $line = $file_handle->getline();
    $line =~ s/^\s+//;  ##remove possible inicial white space
    $line =~ s/^\s+$//; ##remove possible final white space
    $line =~ s/^\n+//;  ##remove possible inicial white line
    $line =~ s/\n+$//;  ##remove possible final white line
    if($line ne "") {   ## do not accept empty line
       push(@gbl_terms,$line);
    }
  }
  $file_handle->close();
}

#create list of terms´s pair
sub PairList() {
       my $file = "pair_list.txt";
  chomp($file);
  my $file_handle = new FileHandle();
  if (!$file_handle->open(">$file")) {
    die("Could not open output file '$file'\n");
  }
       for(my $ii = 0; $ii <= $#gbl_terms; $ii++) {
       for(my $jj = $ii + 1; $jj <= $#gbl_terms; $jj++) {
              #keeping the pair on a hash
              $pair_list{$gbl_terms[$ii]." , ".$gbl_terms[$jj]}++;
              #printing the pair to the pair_list.txt file
              print $file_handle "$gbl_terms[$ii] , $gbl_terms[$jj]\n";
       }## for $jj
       }      ## for $ii
       $file_handle->close();
```

```
}

sub Phrases() {
  my $file = "phrase.txt";
  chomp($file);
  my $file_handle = new FileHandle();
  if (!$file_handle->open($file)) {
    die("Could not open input file '$file'\n");
  }
  while (!$file_handle->eof()) {
    my $line = $file_handle->getline();
    $line =~ s/^\s+//;  ##remove possible inicial white space
    $line =~ s/^\s+$//; ##remove possible final white space
    $line =~ s/^\n+//;  ##remove possible inicial white line
    $line =~ s/\n+$//;  ##remove possible final white line
    #chomp($line);
      push(@gbl_phrase,$line);
  }
  #print @gbl_phrase;
  $file_handle->close();
}

sub Init() {
       for(my $ii = 0; $ii <= $#gbl_terms; $ii++) {
              if(!defined($gbl_term_index{$gbl_terms[$ii]})) {
                     @{$gbl_term_index{$gbl_terms[$ii]}} = ();
              }
       }
}

sub IndexList() {
  my $filename_out = "index_list.txt";
  my $file_handle = new FileHandle();
  if (!$file_handle->open(">$filename_out")) {
    die("Could not open input file '$filename_out'\n");
  }
  Init();
  for(my $ii = 0; $ii <= $#gbl_phrase; $ii++) {
```

```
        for(my $jj = 0; $jj <= $#gbl_terms; $jj++) {
                if($gbl_phrase[$ii] =~ /$gbl_terms[$jj]/) {
                        push(@{$gbl_term_index{$gbl_terms[$jj]}},$ii);
                } ##if
        }       ##for $jj
  } ##for $ii
  for(my $ii = 0; $ii <= $#gbl_terms; $ii++) {
        #for(my $jj = 0; $jj <= $#{$gbl_term_index{$gbl_terms[$ii]}}; $jj++) {
                print $file_handle "$gbl_terms[$ii] -
@{$gbl_term_index{$gbl_terms[$ii]}}\n";
        #}
  }
}

sub InitInvert() {
        for(my $ii = 0; $ii <= $#gbl_terms; $ii++) {
                for(my $jj = 0; $jj <= $#{$gbl_term_index{$gbl_terms[$ii]}};
$jj++) {

        if(!defined($gbl_index_terms{${$gbl_term_index{$gbl_terms[$ii]}}[$jj]})
) {

        @{$gbl_index_terms{${$gbl_term_index{$gbl_terms[$ii]}}[$jj]}} = ();
                        }
                }
        }
}
my @temp = ();

sub InvertIndexList () {
        my $filename_out = "invert_index_list.txt";
  my $file_handle = new FileHandle();
  if (!$file_handle->open(">$filename_out")) {
    die("Could not open input file '$filename_out'\n");
  }

  InitInvert();
  for(my $ii = 0; $ii <= $#gbl_terms; $ii++) {
```

```
        for(my $jj = 0; $jj <= $#{$gbl_term_index{$gbl_terms[$ii]}}; $jj++) {


        push(@{$gbl_index_terms{${$gbl_term_index{$gbl_terms[$ii]}}[$jj]}},"
$gbl_terms[$ii] ");

        #push(@{$gbl_index_terms{${$gbl_term_index{$gbl_terms[$ii]}}[$jj]}},"\t
");
        }
  }
  my @keys = sort({$gbl_index_terms{$b} <=> $gbl_index_terms{$a}} keys
%gbl_index_terms);
  foreach my $key (@keys) {
        #printing to the invert_index_list.txt file
    print $file_handle "$key - @{$gbl_index_terms{$key}}\n";
    #making the co-ocorrency between the list elementes
    for(my $ii = 0; $ii <= $#{$gbl_index_terms{$key}}; $ii++) {


              for(my $jj = 0; $jj <= $#{$gbl_index_terms{$key}}; $jj++) {

                      if($ii != $jj) {
                              #keeping the pair on a hash
                              $pair_list{${$gbl_index_terms{$key}}[$ii]." |
".${$gbl_index_terms{$key}}[$jj]}++;
                              #making the co-ocorrency index list


        push(@{$co_ocorr_index{${$gbl_index_terms{$key}}[$ii]}},${$gbl_index_te
rms{$key}}[$jj]);
                      }
              }## for $jj
              }       ## for $ii
  }
  $file_handle->close();
}

sub CoOcorrPair() {
```

```
	my $filename_out = "co_ocorrency.txt";
  my $file_handle = new FileHandle();
  if (!$file_handle->open(">$filename_out")) {
    die("Could not open input file '$filename_out'\n");
  }
  my @keys = sort({$pair_list{$b} <=> $pair_list{$a}} keys %pair_list);
	foreach my $key (@keys) {

		print $file_handle "$key - $pair_list{$key}\n";
  }
  $file_handle->close();
}

sub CoOcorrIndex() {
	my $filename_out = "co_ocorr_index.txt";
  my $file_handle = new FileHandle();
  if (!$file_handle->open(">$filename_out")) {
    die("Could not open input file '$filename_out'\n");
  }

  my @keys = sort({$co_ocorr_index{$b} <=> $co_ocorr_index{$a}} keys
%co_ocorr_index);
	foreach my $key (@keys) {
		my %temphash = ();
		for(my $ii = 0; $ii <= $#{$co_ocorr_index{$key}}; $ii++) {
			$temphash{${$co_ocorr_index{$key}}[$ii]}++;
		}
	@temp = sort({$temphash{$b} <=> $temphash{$a}} keys %temphash);
		print $file_handle "$key - ";
		foreach my $key (@temp) {
			print $file_handle "$key($temphash{$key})  ";
		}
		#print $file_handle %temphash;
		print $file_handle "\n";
  }
  $file_handle->close();
}
```

```
Terms();
Phrases();
IndexList();
InvertIndexList ();
CoOcorrPair();
CoOcorrIndex();


exit(0);
```

Anexo B

Practical Extraction & Report Language

Tutorial

Índice de conteúdos

1. Introdução

Perl é uma linguagem interpretada optimizada para digitalização arbitrária ficheiros de texto, extrair informações provenientes dos mesmos, e imprimir relatórios com base nessa informação.
Um programa Perl consiste de um grupo de instruções e chamadas de funções da linguagem Perl gravados num ficheiro ASCII (texto) cuja extensão é **.pl** Quando se trata de **CGI**, muitos servidores na Web podem exigir que a extensão seja **.cgi** ou **.pl** e que o programa resida num directório especial chamado **cgi-bin**. A primeira linha de um programa Perl deve ser a directiva **`#!/usr/local/bin/Perl`** para informar o sistema onde encontrar o interpretador Perl.

2. Variáveis e Tipos de Dados

As variáveis são nomes de lugar onde é armazenada uma informação. Em Perl não há necessidade de fazer declaração de variáveis. Uma variável é gerada dinamicamente quando a referenciamos. As variáveis podem ser de três tipos: **escalares** , **arrays** e **Hash** (arrays associativos).

2.1. Escalares

As variáveis escalares começam com um símbolo **$**(cifrão) seguido de uma sequência de caracteres, como **$data**, **$idade**, **$nome**, etc. Deve-se evitar o uso de variáveis como **$1**, **$2**, etc, pois em Perl elas têm significado especial. Além disso, os escalares podem ser do tipo inteiro, ponto flutuante ou string, mas não é necessário especificar isso em Perl pois o interpretador se incumbe de gerar o tipo adequado de acordo com o contexto.

```
$nome = "Luís Barreto";   # define um escalar string "Luís Barreto"
$contador = 312;          # define um escalar inteiro 312
$contador = 3.12;         # define um escalar em ponto flutuante 3.12
```

2.2. Listas e Arrays

Uma lista é uma sequência ordenada de escalares, onde cada elemento tem uma posição fixa, ou um índice, enquanto que um array é um tipo de variável que suporta a lista.

As variáveis do tipo array começam com o símbolo @(arroba). Em Perl, ao contrário de muitas outras linguagens, há um cuidado com a memória dos arrays, ou seja, os arrays crescem e diminuem automaticamente, à medida que se vai introduzindo ou retirando dados. Por exemplo, se inserirmos um escalar numa posição qualquer, por exemplo 10, o Perl automaticamente alarga o array para 10 posições, atribuindo um valor **undef** para as posições não inicializadas no intervalo 0 a 9.

Uma lista vazia é representada por um par de parênteses vazios, como mostram os seguintes exemplos.

```
@vazia = ();                                   # lista vazia
@inteiros = (1,2,3);                           # três números
@string = ("Luís","Carlos","Pedro");           # três strings
@mista = ("pi",3.14,46,"20-03-1977");          # lista mista
```

Não só um array pode manter uma lista de escalares, ele pode também dar-nos o acesso a cada um dos elementos individualmente através da sua posição na lista, através do operador "[ ]". o índice do primeiro elemento da lista é o zero, $array[0]. Aqui nós usamos o $ em vez do @ porque um elemento duma lista é um escalar. Pode-se também aceder aos elementos duma lista por selecção, como @array[1,4,6] ou @array[1..5], a que se dá o nome de slice, e neste caso não é usado o $ porque o resultado é uma lista.

Uma outra forma de saber o número de elementos duma lista é através da notação especial $#array, dá o índice do último elemento do @array. Como exemplo:

```
@array  = ("Luís","Carlos","Pedro","Barreto");
$length = @array;
print $length;              # imprime 4
print $#array;              # imprime 3
print $array[$#array]       # imprime "Barreto"
```

contudo pode-se fazer atribuições às listas como por exemplo:

```
($a, $b) = ("um","dois");
($luís, @outros) = (25,23,32,25,22,26)       # $luís = 25
                                             # e $outros = (23,32,25,22,26)
($array[0],$array[1]) = ($array[1],$array[0]); # troca o primeiro com o segundo
```

Agora vamos ver alguns métodos de representar os conteúdos dum array como uma string ou imprimi-los.

As variáveis de tipo array são interpolados em double quoted strings como numa variável do tipo escalar. A título de exemplo,

@array = ("Luís","Carlos","Pedro","Barreto");
print "O conteúdo do array é $array[0] $array[1] $array[2] $array[3]";

#imprime O conteúdo do array é Luís Carlos Pedro Barreto

também pode-se fazer uma interpolação directamente sobre todo o array:

print " O conteúdo do array é @array";

cujo resultado será o mesmo que no caso anterior. Contudo podemos usar a função join para separar os itens, como exemplo:

$string = join " " , @array; # $string = "Luís Carlos Pedro Barreto"

2.3 hash (arrays associativos)

Os hashes são semelhantes aos arrays do ponto de vista de serem conjunto de escalares onde podem ser acedido através dum índice, com a diferença de que os índices ou chaves podem ser qualquer tipo de escalar, como por exemplo uma string. Numa variável do tipo hash o nome é antecedido do símbolo % (percentagem). Para aceder ao valor dum hash através da sua chave usa-se as chavetas na seguinte notação: `$hash{$chave}`.

A título de exemplo para usar a idade de diferentes pessoas podemos usar os seus nomes como chave para definir um hash.

%idade = (“Luís”,26,”Carlos”,18,”Pedro”,3,”Barreto”,25,);

print "A idade do Luís é $idade{'Luís'}!\n"; # A idade do Luís é 26
Os hashes são criados de duas formas diferentes, em que a primeira atribui-se um valor à chave nomeada,
 $idade{Luís} = 26;

E a segunda forma consiste em usar uma lista onde os elementos desta lista é vista aos pares, em que o primeiro elemento deste par é usada como a chave e o segundo como valor. Por exemplo,

%hash = ('Luís' , '26', 'Carlos', '18');

também pode-se usar => para indicar o par chave/valor,

%hash = ('Luís' => '26', 'Carlos' => '18');

Podemos extrair elementos de forma individual,

print $hash{'Luís'}; # imprime 26

ou então de forma colectiva, recorrendo à função foreach() como o uso da chave,

```
%idade = ("Luís",26,"Carlos",18,"Pedro",3,"Barreto",25);

foreach $chave (%idade) {
  print "A idade do $chave é $idade{$chave} \n";   # A idade do 26 é
}                                                   # A idade do Pedro é 3
                                                    # A idade do 3 é
```

contudo o resultado não é bem o que se estava. Isto deve-se ao facto de que %idade retorna uma lista de valores. Uma forma de resolver este problema é recorrer à função **each** que retorna o par chave/valor.

```
%idade = ("Luís",26,"Carlos",18,"Pedro",3,"Barreto",25);

while (($chave, $valor) = each %idade) {

  print "A idade do $chave é $idade{$chave}\n";  # A idade do Carlos é 18

}                                                      # A idade do Luís é 26
                                                       # A idade do Pedro é 3
```

caso queiramos aceder a um par chave/valor que não existe o que se obtém é um valor indefinido, e caso esteja usando o warnings é lhe notifica sobre isso. Uma forma de ultrapassar isto é fazendo um teste do par usa a função **exists(),** que retorna true caso a chave nomeada exista, ou false caso contrário.

```
%idade = ("Luís",26,"Carlos",18,"Pedro",3,"Barreto",25);

if (exists($idade{"Frederico"}))
{
```

```
  print "Frederico, idade $idade{'Frederico'}!\n";
}
else
{
  print "Frederico, idade desconhecido!\n"; # Frederico, idade desconhecido!
}
```

}

2.3.1. Classificação/Ordenação de Hashes

Contudo não há uma garantia de que a ordem ao qual a lista das chaves, valores, ou o par chave/valor será sempre a mesma.

```
print(join(', ',chaves %hash),"\n");
```

Uma forma de garantir é recorrer ao uso da função sort,

```
print(join(', ',sort chaves %hash),"\n");
```

Nota:

chaves - hash retorna uma lista com apenas as chaves no hash. Como acontece com qualquer lista, utilizando-o num contexto escalar retorna o número de chaves nessa lista.

Valores – HASH retorna uma lista com apenas os valores no hash, na mesma ordem como as chaves são retornadas pelas chaves.

Podemos agora fazer uma lista ordenada a partir do hash.

```
foreach $chave ( sort chave %hash ) {
  push @sortedlist, ( $chave , $hash{$chave} );
  print "A chave $chave tem o valor $hash{$chave}\n";
}
```

este exemplo também mostra que os itens individuais podem ser interpolados em strings double-quoted. Contudo não há maneira de interpolar todo o hash, ou seja, é preciso fazer uma iteração sobre ele ou atribui-lo a uma lista.

As vezes é preciso adicionar ou apagar elementos num hash, a título de exemplo.

```
foreach $ chave ( chaves %hash ){
   if($del_condition) {
      delete $hash{$chave};
   }
   if($add_condition) {
      $hash{$qqcoisa} = $qqcoisa;
   }
}
```

3. Comentários

À semelhança de outras linguagens, também em Perl sentimos necessidade de comentar linhas ou blocos de instruções, etc. para isso é usado o caracter cardinal (#) que para o interpretador Perl qualquer palavra vista depois do # é considerado como um comentário.

4. Ficheiros

Um ficheiro é um conjunto de dados armazenados num disco. Entretanto os ficheiros, para este contexto, serão ficheiros que contêm texto, ou seja, ficheiros que contêm caracteres. Contudo um texto contem mais informações do que simples caracteres, palavras e frases. O texto pode conter informações sobre a sua estrutura ou a forma como as palavras ou frases devem ser expostas, ou sobre o tamanho do caracter e o tipo de letra que deve ser usado para estas letras. Contudo às vezes podemos querer remover toda estas informações do texto antes do seu processamento, mesmo sabendo que estas informações podem ser úteis a nível de análise do texto.

Até agora viu-se três tipos de variáveis: escalares, listas, e hashes. Vamos agora conhecer um novo tipo de tipo de variável, o file Handle. O file handle é definido com comando de abertura em que no caso de estarmos a programar um ambiente strict , eis alguns exemplos usando a abertura:

open(handle,"nome do ficheiro");

open(FILE_HANDLE,"<texto.txt");	**FILE_HANDLE** é o **Handle** do ficheiro o caracter **<** antes do nome do ficheiro, significa que o ficheiro está sendo aberto para leitura. **texto.txt** é o nome do ficheiro. Para ficheiros residentes em outros directórios, diferentes daquele onde se encontra o programa Perl, deve-se especificar o path completo.

Após utilizarmos o ficheiro, este deve ser fechado com a função **close**.

close(FILE_HANDLE);

Do mesmo modo que especificamos o caracter **<** à frente do nome do ficheiro, para dizer que ele será aberto para leitura, outros caracteres são usados para abrir ficheiros para serem gravados ou ampliados. A tabela a seguir mostra essas variações:

Caracter	Significado	Exemplos e comentários

<	Abre ficheiro para leitura.	open(*FILE1*,"<texto.txt");
		ou
		open(FILE1,"texto.txt");
		
		Não é necessário especificar o caracter **<**. Qualquer ficheiro aberto sem a especificação de modo será aberto para leitura.

>	Abre ficheiro para gravação.	open(FILE1,">texto.txt"); Se o ficheiro já existir, ele será regravado e seus dados anteriores serão perdidos. Se o ficheiro não existir, um novo ficheiro será criado.
>>	Abre ficheiro para ampliação.	open(FILE1,">>texto.txt"); Se o ficheiro já existir, os novos dados serão inseridos no final desse ficheiro. Se o ficheiro não existir, um novo ficheiro será criado.
+<	Abre ficheiro para leitura e gravação.	open(FILE1,"+<texto.txt"); Pode-se ler e gravar no mesmo ficheiro.
+>	Abre ficheiro para leitura e gravação.	open(FILE1,"+>texto.txt"); Pode-se ler e gravar no mesmo ficheiro.

4.1 Leitura

Para ler um ficheiro, Perl tem um operador especial que consiste em colocar o **Handle** entre os caracteres **<** e **>** deste modo: **<FILE_HANDLE>**. Podemos, então, atribuir a uma variável escalar ou a um array os registos provenientes de um ficheiro. Veja os exemplos abaixo:

$registo = <FILE_HANDLE>	Lê um registo do ficheiro e o coloca na variável **$registo**
@matriz = <FILE_HANDLE>	Lê todos os registos do ficheiro, **de uma só vez**, e armazena-os no array **@matriz**. Após isso, pode-se obter os registos usando-se o método estudado para os arrays.

quando abrimos um ficheiro temos de definir qual a acção a actuar sobre o ficheiro

o nome do file handle pode ser qualquer coisa, desde que não sejam utilizados caracteres especiais e não necessitam de prefixos especiais ao contrário dos outros tipos de variáveis.

Um **Handle** é, na verdade, um ponteiro para o ficheiro que Perl deve ler ou gravar.

4.2. ficheiros predefinidos

Perl possui três **Handles** especiais, predefinidos:

STDIN	Entrada padrão (teclado)
STDOUT	Saída padrão (monitor de vídeo)
STDERR	Mensagens de erro (monitor de vídeo)

Toda vez que se usa o comando `print ".....";` sem que se especifique o Handle de ficheiro, o resultado será enviado a **STDOUT**, o mesmo se dando com o comando `read` que lerá obterá a entrada do teclado **STDIN**.

4.3. Imprimir num ficheiro novo

A impressão num novo ficheiro implica em usarmos o caracter **>** na frente do nome do ficheiro durante sua abertura. Além disso, para gravarmos um ficheiro, usamos a função **print** usando como primeiro argumento o **Handle** do ficheiro.

A instrução **foreach** é especialmente usada para operações com **arrays**. Do mesmo modo que a instrução **for**, ela executa as declarações entre as chaves.
No exemplo acima

```
@cores = ("azul","verde","amarelo","vermelho");
foreach $cor (@cores) {
    print $cor,"\n";
}
```

Na primeira vez, a variável **$cor** assume o valor "**azul**" que será impresso a seguir. Na segunda vez, assume o valor "**verde**" e assim por diante. Não é necessário que o programador se preocupe com o número de elementos do array porque **foreach** faz isso automaticamente.

Outra maneira de se usar a instrução **foreach** é através de um indexador numérico como no exemplo abaixo:

```
@cores = ("azul","verde","amarelo","vermelho");
foreach $i (0..$#cores) {
     print $cores[$i],"\n";
}
```

Nesse exemplo, a variável **$i** assumirá todos os valores, de **zero**, que é o número do primeiro elemento do array, até **3**, que é o número do último elemento.

4.4. Ampliar um ficheiro existente

A ampliação de um ficheiro, implica a utilização dos caracteres **>>** à frente do nome do ficheiro durante sua abertura. Para gravarmos novos registos no ficheiro. Como exemplo, vamos ampliar o ficheiro do exemplo anterior:

```
open(CORES,">>cores.txt");                    # abre ficheiro para ampliação
@cores = ("branco","rosa","violeta","laranja");   # gera array com os nomes das
cores
```

```
foreach $i (@cores)                    # para cada cor no array
{                                      # inicio do bloco foreach
print CORES $cores[$i],"\n";           # grava um registo no ficheiro
}                                      # fim do bloco foreach
close(CORES);
```

5. Operadores Relacionais

Os operadores relacionais são símbolos usados entre os operandos nas expressões condicionais. Em Perl existe uma diferença entre o contexto numérico e o contexto de string no que diz respeito ao uso dos operadores relacionais, isto é, uma comparação entre duas variáveis numéricas deve ser feita da seguinte maneira: **if($x == $y).....** enquanto uma comparação entre duas variáveis string seria: **if($a eq $b).....** A tabela abaixo mostra todos os operadores relacionais para serem usados em ambos os contextos:

Operador	**Contexto numérico**	**Contexto string**
Igualdade	==	eq
Desigualdade	!=	ne
Maior do que	>	gt
Maior ou igual a	>=	ge
Menor do que	<	lt
Menor ou igual a	<=	le

6. Operações com strings (substituição e correspondência)

Em Perl há uma grande necessidade de alterar parte de uma string. Para isso o Perl disponibiliza dois operadores o **s/ / /** que modifica sequências de caracteres e o **tr/ / /** que modifica os caracteres individualmente, contendo duas partes ambos os operadores, em que a primeira parte entre as duas primeiras barras contem um search parttern e segunda parte entre as duas últimas barras é a parte de substituição. À frente da última barra podemos pôr caracteres que modificam o comportamento dos comandos, em que por defeito s/ / / apenas substitui a primeira ocorrência do search pattern e quando pomos um **g** o operador substitui todas as ocorrências. O operado **tr/ / /** permite a modificação dos caracteres, **c** (substitui o complemento da search class), **d** (apaga os caracteres da search class que não estão substituídas) e **s** (substitui sequências idênticas de caracteres por um caracter). Eis alguns exemplos:

```
$text =~ s/bug/feature/; # substitui a primeira ocorrência de "bug" por  "feature"

$text =~ s/bug/feature/g; #substitui todas as ocorrências de "bug" por  "feature"

$text =~ tr/[A-Z]/[a-z]/; #converte para minúsculos

$text =~ tr/AEIOUaeiou//d; # apaga todos os vogais

$text =~ tr/[0-9]/x/cs; # substitui a sequência não numérica por x

$text =~ s/[A-Z]/CAPS/g; # substitui todos os caracteres em maiúsculo por CAPS
```

Alguns caracteres especiais

- \b: word boundaries (palavra fronteiras)
- \d: dígitos
- \n: nova linha
- \r: carriage return
- \s: espaço em branco
- \t: tab
- \w: caracteres alfanuméricos
- ^: início da string

- `$`: fim da string
- `.`: qualquer caracter
- `[bdkp]`: caracteres b, d, k e p
- `[a-f]`: caracteres de a à f
- `[^a-f]`: todos os caracteres excepto de a à f
- `abc|def`: string abc ou string def

Nota:

\w corresponde às sequências a-zA-Z0-9 o que não corresponde aos caracteres com acento ou caracteres especiais, ou então depende da configuração do sistema operativo.

Com estas sequências podemos criar muitas expressões regulares, mas continuamos a precisar de mais, como por exemplo qualquer coisa para encontrar a repetição de uma mesma string com comprimento arbitrário, por exemplo ha haha, hahaha e assim por diante. Isto é possível com os quantificadores. Eis os quantificadores pelas expressões regulares do Perl:

- `*`: zero ou mais vezes
- `+`: uma ou mais vezes
- `?`: zero ou uma vez
- `{p,q}`: pelo menos p vezes e no máximo q vezes
- `{p,}`: pelo menos p vezes
- `{p}`: exactamente p vezes

os quantificadores operam sobre o anterior token. Por exemplo, *ha** corresponderá *h, ha, haaa,* e assim por diante. Se quisermos operar sobre mais do que um caracter temos de incluir estes caracteres entre parênteses. Sendo assim, por exemplo se aplicarmos (ha)+ sobre "*ele disse hahaha"* corresponderá ao *hahaha.*

7. Sub-rotinas

Uma sub-rotina é uma espécie de função escrita pelo próprio programador e que executa uma tarefa repetitiva dentro do programa. A função de uma sub-rotina é executar trechos de códigos repetitivos, que deveriam ser codificados várias vezes dentro dos programas ou accionados através da instrução **goto**. As sub-rotinas podem receber valores e também devolver valores ao seu chamado.

Em Perl, define-se uma sub-rotina em qualquer lugar dentro de um programa usando-se o identificador **sub** seguido do nome da sub-rotina. Uma subrotina tem o seguinte aspecto:

```
sub NomeSubRotina {
  declarações;
}
```

O exemplo abaixo mostra uma sob-rotina completa. Note que a sub-rotina é chamada colocando-se o caracter **&** como prefixo de seu nome.

```
#-------------------------------------------------------#
#          Exemplo de sub-rotina                  #
#-------------------------------------------------------#
for($n = 0;$n <= 5;$n++) {
   &frase;
}
sub frase {
   print "Exemplo de sub-rotina\n";
}
```

7.1. Retornando valores

As sub-rotinas podem devolver valores ao programa principal. Talvez seja esta a maior utilidade das sub-rotinas. Os valores retornados podem ser escalares ou arrays. Devemos tomar muito cuidado na hora de codificar uma sub-rotina que retorna um valor pois, o valor retornado é sempre o valor da **última** declaração dentro da sub-rotina. Portanto, se a última declaração de uma sub-rotina for um `print` , ela retornará o valor **1**.

Exemplo

```
#-----------------------------------------------------#
#              Exemplo de sub-rotina              #
#-----------------------------------------------------#
$x = &valor;
if($x == 1) {
    print "Valor retornado = ",$x,"\n";
 }
 sub valor {
    print "Esta sub-rotina retorna um valor\n";
}
```

7.2. Passagem de parâmetros

As vezes é necessário passar certos valores para uma sub-rotina, como números, a fim de fazer cálculos e obtermos os resultados. Os valores a serem passados são informados como se fossem listas de variáveis entre parênteses na chamada da sub-rotina. A lista de variáveis informadas na chamada, chegam para a sub-rotina no array especial @_. Isso significa que Perl nunca trabalhará directamente com as variáveis originais, mas com cópias dessas variáveis.

Exemplo

```
#-----------------------------------------------------#
#           Exemplo de sub-rotina                 #
#-----------------------------------------------------#
$x = &area(18,24);
print "Area = ",$x,"\n";

sub area {
   local($a,$b) = @_;
   return($a * $b);
}
```

No exemplo acima, a função da sub-rotina `&area` é calcular a área de qualquer rectângulo. Na sua chamada foram passados os valores **18** e **24**, que são as medidas dos lados de um rectângulo. A sub-rotina recebeu os valores 18 e 24 nas variáveis locais **$a** e **$b** através do array especial @_, usando-se para isso a declaração **local** da linguagem Perl.

A seguir, a declaração **return($a * $b)** faz a multiplicação de **$a * $b** devolvendo imediatamente o resultado para a variável **$x**.

Outra versão da mesma sub-rotina é mostrada abaixo. Aqui, deve-se notar que a sub-rotina pode ser chamada de dentro da própria função "print". O exemplo abaixo calcula as áreas de todos os rectângulos cujos lados variam de 1 a 4.

Exemplo

```
#-------------------------------------------------------#
#             Exemplo de sub-rotina              #
#-------------------------------------------------------#
for($x = 1;$x <= 4;$x++) {
    for($y = 1;$y <= 4;$y++) {
    print "Área de ",$x," X ",$y," = ",&area($x,$y),"\n";
      }
}

sub area {
      local($a,$b) = @_;
      return($a * $b);
}
```