Dissertação/Estágio

Mestrado em Engenharia Informática

Pesquisa Semântica Baseada em Contexto Móvel

António Fernando Santos Marques

antoniom@student.dei.uc.pt

Orientador

Paulo Jorge de Sousa Gomes

Departamento de Engenharia Informática

Faculdade de Ciências e Tecnologia

Universidade de Coimbra

2013/2014

Resumo

Com a grande utilização dos *smartphones* nas nossas tarefas diárias, a quantidade de informação do utilizador que é processada deve ser levada em conta no desenvolvimento de aplicações que se baseiam neste tipo de dados.

Dentro das várias aplicações que se podem desenvolver a partir desta informação, surgem os mecanismos de pesquisa, sendo que no ramo de aplicações orientadas ao ambiente móvel, a pesquisa baseada em contexto do utilizador, é um tipo de solução que ainda está em fase de crescimento. Com base nisso, é possível explorar como é que as pesquisas de um utilizador podem ser adaptadas às suas necessidades, a fim de resolver o problema das listas de resultados extensas que muitas vezes são apresentadas, assim como o facto de, por vezes não se conseguir encontrar aquilo que se procura.

Atendendo ao problema, nesta dissertação desenvolveu-se uma aplicação para dispositivos *Android*, que baseada na informação diária do utilizador (mensagens de texto, *emails*), procedeu à extração de interesses do mesmo para que fosse construída uma ontologia representativa do perfil do utilizador. Posteriormente, foi implementado um mecanismo de pesquisa na aplicação que atuou sobre um *dataset* da DBpedia e utilizou o perfil do utilizador para influenciar a apresentação dos resultados. Para além da aplicação móvel, também foi desenvolvida uma aplicação servidor, que é fundamental para o processamento da informação recolhida. Por fim, foi feita uma experimentação deste protótipo, em que os resultados demonstraram-se positivos permitindo afirmar que a informação extraída de um *smartphone* pode influenciar a apresentação de resultados num motor de busca.

Palavras-Chave: Contexto, Aplicação Móvel, *Smartphone*, *Android*, *Web* Semântica, Pesquisa, Ontologias, Interesse, DBpedia

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

ES *Executor Service*

IA Inteligência Artificial

IR *Information Retrieval*

KIS Knowledge and Intelligent Systems

MSCS *Mobile Semantic Context-Based Search*

PLN Processamento de Linguagem Natural

POS *Part-of-Speech*

RDF Resource Description Framework

REM Reconhecimento de Entidades Mencionadas

WS Web Semântica

Capítulo 1

Introdução

Atualmente, ouvir falar em termos como *"Android"*, *"iPhone"* e *"Smartphone"* já é algo natural no quotidiano das pessoas. Este fator deve-se à evolução que o mundo da tecnologia sofre diariamente, sendo que para o ambiente móvel é desenvolvido cada vez mais *software* que procura auxiliar a sociedade nas suas tarefas.

A pesquisa de informação na *web* é algo que, para maior parte da população, se torna uma tarefa frequente, seja a realizar trabalhos para a escola, seja a procurar atividades de lazer, entre outros fins. Para tal, motores de busca como *"Google"*[1], *"Yahoo!"*[2], *"Bing"*[3] são as ferramentas mais utilizadas para essas pesquisas. No entanto, estes sistemas nem sempre apresentam os resultados esperados, pois não passam por entender aquilo que o utilizador pretende.

Interpretar o objetivo do utilizador ao efetuar uma pesquisa implica que seja conhecido o contexto do mesmo. Desta forma, os *smartphones* acabam por ser uma mais valia no que diz respeito ao processar informação do utilizador, isto porque estes tornam-se uma extensão do mesmo nas suas tarefas diárias, sendo possível explorar características únicas do utilizador.

Assim, o foco desta dissertação passou por explorar uma forma de conseguir efetuar pesquisas baseadas em contexto, sendo este recolhido de informação diária do utilizador. Para isso, foi desenvolvida uma aplicação móvel capaz de extrair informação a partir de várias interações realizadas no *smartphone* (mensagens de texto, *emails*) e assim construir interesses do utilizador. Posteriormente, estes vão ser utilizados para influenciar os resultados das pesquisas efetuadas, de forma a que estes estejam de acordo com a preferência do utilizador. Foi também desenvolvida uma aplicação servidor, que é responsável por tratar toda a informação recolhida do utilizador,

[1] https://www.google.com/
[2] https://www.yahoo.com/
[3] http://www.bing.com/

construindo interesses do mesmo, assim como processar os pedidos de pesquisa, entre outros.

Na implementação deste sistema são abordados vários tópicos como o Processamento de Linguagem Natural (PLN) [1], as tecnologias da Web Semântica (WS) [2], *Information Retrieval* (IR) [3] e o contexto do utilizador, que permitem desenvolver as ferramentas necessárias para a implementação desta aplicação. A privacidade dos dados, como por exemplo, credenciais do utilizador, mensagens de texto, foram levadas em conta, sendo utilizado um mecanismo de encriptação dos mesmos. No entanto, o objetivo da dissertação passa só pela elaboração dos mecanismos de extração e pesquisa.

O protótipo a ser desenvolvido, *Mobile Semantic Context-Based Search* (MSCS) está inserido no âmbito do laboratório Knowledge and Intelligent Systems (KIS), pertencente ao Departamento de Engenharia Informática[4] da Faculdade de Ciências e Tecnologia da Universidade de Coimbra. Este projeto foi inicializado por um investigador do laboratório KIS, David Miranda, que já havia realizado algum trabalho. No entanto, apesar de terem sido mantidas as tecnologias a utilizar, a implementação do sistema foi reformulada.

Para supervisionar o desenvolvimento desta dissertação ficou destacado como orientador, o professor Paulo Gomes do Departamento de Engenharia Informática, que contribuiu com o esclarecimento das várias dúvidas que foram surgindo no seu processo, assim como também contribuiu para elaboração deste documento revendo-o e sugerindo alterações que se tornaram necessárias para o mesmo.

As contribuições esperadas do protótipo desenvolvido são:

- Enriquecimento do estado da arte:
    1. Processamento de Linguagem Natural sobre informação retirada de dispositivos móveis
    2. Construção de uma ontologia de perfil do utilizador a partir de informação recolhida do *smartphone*, assim como um mecanismo de pesquisa baseado na mesma, remetendo para as tecnologias da *Web* Semântica
    3. Na área de *Information Retrieval*, este trabalho explora a associação de informação recolhida com o perfil do utilizador, assim como com um repositório da DBpedia
- Protótipo capaz de efetuar pesquisas baseadas em contexto do utilizador
- Experimentações feitas ao protótipo, com vários utilizadores, que consigam demonstrar a influência do contexto nas pesquisas efetuadas

[4] http://www.uc.pt/fctuc/dei/

- Documento de dissertação que descreve todo o trabalho realizado para o desenvolvimento do protótipo

De forma a ser melhor entendido os tópicos que vão ser abordados ao longo deste documento, procede-se a uma descrição do que é descrito em cada um dos capítulos desta dissertação.

No capítulo 2, são descritos os conceitos do estado da arte que devem ser compreendidos para um bom entendimento desta dissertação, tais como, Processamento de Linguagem Natural, tecnologias da *Web* Semântica e *Information Retrieval*. São ainda apresentados trabalhos que podem ser relacionados com o que é pretendido para esta dissertação, no sentido de conseguir obter uma percepção de resultados que já foram possíveis atingir dentro desta área.

No capítulo 3, estão descritas as especificações do sistema que se pretende desenvolver, tais como os casos de uso, requisitos do sistema, tecnologias que são utilizadas para o desenvolvimento do mesmo e representações arquiteturais do mesmo.

De seguida no capítulo 4, é apresentado o sistema desenvolvido para esta dissertação, no sentido de clarificar como é este se encontra estruturado, assim como cada operação que teve de ser realizada para atingir o objetivo do projeto.

No capítulo 5 está especificado como é que foi feita a avaliação do protótipo, isto é, as fases em que esta se dividiu, a explicação dos testes efetuados e os resultados obtidos.

O capítulo 6 demonstra o planeamento estruturado para esta dissertação. Aqui é avaliado o cumprimento de objetivos durante os dois semestres, no sentido de analisar as diferenças sobre o que se encontrava estipulado e aquilo que foi realmente cumprido.

Por fim, no capítulo 7 é feita a conclusão desta dissertação. Esta passa por uma análise crítica sobre o sistema desenvolvido e os seus resultados, assim como é apresentado trabalho que poderá ser implementado neste protótipo no futuro.

Capítulo 2

Estado da Arte

No presente capítulo são apresentados os conceitos fundamentais para a compreensão desta dissertação. É feita uma abordagem ao Processamento de Linguagem Natural (PLN), que se torna fundamental para processos sobre o texto extraído, assim como para desambiguação deste. As tecnologias da Web Semântica (WS) entram no sentido de obter a informação estruturada e partilhada. De seguida é apresentada uma breve definição de contexto, visto que é um ponto fundamental desta dissertação e que deve ser analisado aos diversos níveis. De seguida é apresentado o tópico de *Information Retrieval* (IR), que é relevante para a recolha de informação adaptada ao contexto do utilizador. Por fim, são apresentados alguns trabalhos que se encontram relacionados com o que é realizado nesta dissertação.

2.1 Processamento de Linguagem Natural

O Processamento de Linguagem Natural (PLN) [1] é uma área da Inteligência Artificial (IA) [4] que visa construir mecanismos de comunicação entre as pessoas e as máquinas. Ao longo dos anos, este tipo de processamento foi várias vezes apresentado como algo futurista, como por exemplo, na série televisiva *Knight Rider*[1] em 1982. Nesta, o protagonista interagia com o seu carro através de comunicação por voz, utilizando linguagem natural e este respondia através de linguagem natural. No entanto, este exemplo trata-se de um tipo de comunicação que não vai ser abordada durante esta dissertação. Um caso de muito sucesso relativamente ao PLN é o *Cleverbot*[2]. Este despertou a curiosidade de muitas pessoas pelo facto de ser possível comunicar com o computador de uma forma natural.

[1] http://www.imdb.com/title/tt0083437/
[2] http://www.cleverbot.com/

Esta área da IA está a crescer cada vez mais, pois existe uma necessidade de as pessoas conseguirem comunicar com as suas máquinas de uma forma mais natural, temos o caso do Siri[3] da Apple[4] e também outro tipo de interações como resultados de pesquisas que correspondam, de uma forma ainda mais precisa, ao que utilizador pretende.

No entanto, ainda existem alguns problemas no que diz respeito ao processamento de linguagem natural, mais precisamente o fato de as linguagens naturais terem ambiguidade a vários níveis (palavras com mais de um significado, contexto da frase), enquanto que a linguagem dos sistemas informáticos (ex. linguagens de programação) são não ambíguas.

Para esta dissertação vamos concentrar-nos apenas nos três níveis que são mais apropriados, o morfológico, que visa determinar a categoria gramatical de cada palavra, o sintático, para conseguir relacionar as palavras entre si e o semântico, para determinar o sentido da linguagem. Estes níveis são descritos nas seguintes subsecções e é demonstrado, com exemplos, como a ambiguidade surge em cada um dos casos.

2.1.1 Análise Morfológica

A análise morfológica destina-se à avaliação individual das várias palavras constituintes de uma frase e, desta forma, classificá-las de acordo com as suas categorias gramaticais e morfológicas. Nesta análise, faz-se o estudo da estrutura da palavra, e assim é possível determinar os lexemas e morfemas que lhe dão origem, os seus processos de formação (derivação, composição, etc) e outras características dependendo da sua categoria (género, número, etc).

São apresentadas algumas técnicas utilizadas durante a análise morfológica:

- ***Tokenization***: Este processo destina-se a dividir texto em várias unidades, denominadas por *tokens*, podendo estes ser palavras, números, sinais de pontuação ou qualquer outra estrutura. A partir desta divisão, essas unidades são então analisadas e classificadas. No entanto, surgem alguns problemas na utilização desta técnica, tais como, os dois pontos ":", que podem ser utilizados para demonstrar as horas ou então o ponto "."que pode servir para abreviar palavras, entre outras coisas. Um exemplo que se adequa perfeitamente a esta dissertação, é a utilização dos *emoticons* como por exemplo ":-)", pois como o contexto do utilizador é baseado em *SMS's* e *emails*, é natural aparecerem os mesmos. Nestes casos, agrava-se a situação no que diz respeito à divisão por *tokens*. Outro caso popular é a utilização dos espaços em branco como delimitadores dos *tokens*. Isto, em casos

[3]http://www.apple.com/ios/siri/
[4]http://www.apple.com/

que queremos reconhecer palavras do tipo "Engenharia Informática", é necessário alterar o processo de divisão, de maneira a que sejam reconhecidas como uma única palavra.

- ***Sentence Splitting***: Esta técnica acaba por ser uma extensão da *tokenization*, em que são identificadas várias frases a partir dos seus elementos de pontuação, como por exemplo pontos finais, de exclamação, interrogação, etc.

- ***Spell Checking***: No PLN, é necessário proceder à validação de erros de ortografia. Quando estamos a fazer verificação, esta divide-se em três passos principais. O primeiro passa por analisar o texto ou excerto e extrair as várias palavras deste. De seguida, cada uma destas palavras é comparada com uma lista/dicionário de palavras de forma a validá-las. Se estas palavras não forem encontradas nessa lista, são marcadas como potencialmente erradas. Nesses casos, procura-se por palavras que são semelhante às potencialmente erradas e faz-se a associação. Um passo extra seria considerar diferentes formas da palavra, para evitar correção de palavras que estariam corretamente escritas.

- **Remoção de *Stopwords***: O objetivo desta técnica é remover palavras que vão acabar por ser pouco específicas para o processo de recolha de informação. As palavras consideradas como *stopwords*, normalmente, são artigos, preposições, conjunções ou então palavras muito comuns da língua. Na figura 2.1 é dado um exemplo de remoção destes termos.

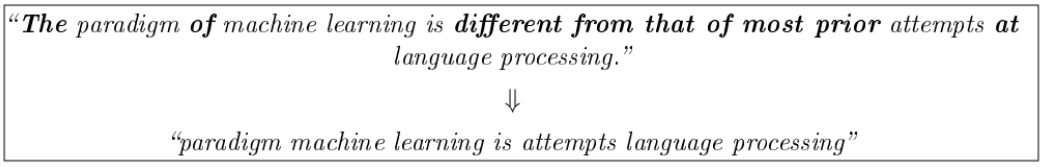


Figura 2.1: Exemplo de remoção de *stopwords*.

- ***Stemming***: Nesta técnica, a redução de uma palavra ao seu *stem* baseia-se em remoção de sufixos das palavras. Este processo não tem conhecimento do contexto, dando origem, por vezes, a palavras sem significado nenhum. No entanto, os *stemmers* são fáceis de implementar, tornando-se mais apropriado para aplicações mais simples. Alguns exemplos de *stemming* são apresentados (ver figura 2.2).

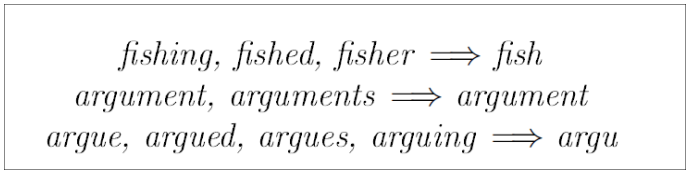


Figura 2.2: Exemplos de redução de palavras ao seu *stem*.

- **Lematização**: Este processo é uma forma de normalizar palavras, removendo terminações das mesmas, de forma a que estas sejam reduzidas à sua forma original, conhecida como

lemma. Ao contrário da técnica de *stemming*, esta tem conhecimento sobre o contexto e conseguem discriminar palavras com diferentes significados. Dando o exemplo da palavra *better*, esta tem como *lemma*, a palavra *good*. Esta redução consegue ser bem sucedida, pois o processo passa por pesquisa em dicionários. Esta técnica é mais complexa que a de *stemming*, sendo por vezes melhor optar por uma abordagem mais simples. Outros exemplos de *lemmatization* são apresentados na figura 2.3.

$$went,\ goes \Longrightarrow go$$
$$computers,\ computing,\ computation \Longrightarrow computer$$

Figura 2.3: Exemplos de redução de palavras ao seu *lemma*.

- **Reconhecimento de Entidades Mencionadas (REM)**[5]: Esta técnica está incluída na área de extração de informação e tem como objetivo reconhecer entidades em textos de linguagem natural. Para tal reconhecimento é necessário seguir um conjunto de passos. O processo é demonstrado pela figura 2.4.

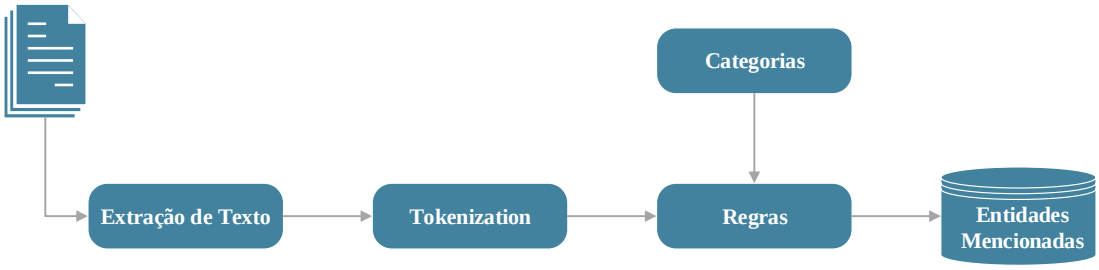


Figura 2.4: Processo de reconhecimento de entidades mencionadas

Após o texto ser extraído do documento, este passa por um processo de *tokenization*, em que posteriormente, os *tokens* gerados vão ser associados a um conjunto de categorias (Produto, Pessoa, Organização, etc..) seguindo um conjunto de regras. Na figura 2.5 é apresentado um excerto de texto, com entidades reconhecidas.

"**Steve Jobs (Pessoa)** lançou o novo **iPhone (Produto)**".

Figura 2.5: Exemplo de frase com entidades reconhecidas e categorizadas.

Relativamente ao tipo de ambiguidade que pode surgir nesta análise, podemos dar como exemplo a classificação da palavra "canto". Esta palavra pode assumir o papel de um substantivo ou de um verbo. No entanto, a análise morfológica apenas lida com a análise, a identificação e descrição da

[5]Conhecida pelo termo em inglês *Named Entity Recognition* (NER)

estrutura interna das palavras. Deste modo, não conseguem detetar em que contexto a palavra se insere, sendo necessário recorrer a uma análise sintática para identificar a forma com as palavras se relacionam na frase.

2.1.2 Análise Sintática

Como foi referido anteriormente, a análise sintática concentra-se no estudo da relação entre palavras numa frase. Aqui as palavras são agrupadas de acordo com a sua função e, a partir da forma como se encaixam na frase e das suas palavras vizinhas, estas podem assumir diferentes categorias. Existem alguns casos em que as palavras vizinhas auxiliam na classificação da palavra, como por exemplo:

- Os pronomes possessivos, normalmente, são seguidos de nomes.
- Os pronomes pessoais, normalmente, são seguidos de verbos.

Dentro da análise sintática, é conhecida uma técnica bastante reconhecida no que diz respeito a classificar gramaticalmente cada palavra, sendo esta a técnica de *Part-of-Speech (POS) Tagging*.

Esta técnica tem como objetivo atribuir a cada uma das palavras no texto uma classe gramatical, como por exemplo, nomes, adjetivos, pronomes, entre outros. Para a atribuição dessas classificações, são analisadas palavras adjacentes e relacionadas com a que se pretende classificar.

Existe já definido um conjunto de *tags* para identificar cada classe. Na tabela 2.1 é possível verificar as principais *tags* de acordo com o projeto *Penn Treebank*[6].

POS Tag	Descrição	POS Tag	Descrição
CC	Coordinating Conjunction	NNP	Proper Noun, singular
CD	Cardinal Number	PRP	Personal Pronoun
DT	Determiner	RB	Adverb
IN	Preposition	SYM	Symbol
JJ	Adjective	VB	Verb, base form
NN	Noun, singular	VBD	Verb, past tense

Tabela 2.1: Exemplos de *tags* para o processo de POS *Tagging*.

Com base nos diferentes tipos de *tags*, podemos observar um exemplo de POS Tagging (ver figura 2.6).

[6]http://www.cis.upenn.edu/~treebank/

"O cão atacou o gato."
O $\Longrightarrow$ Determinante (DT)
cão$\Longrightarrow$ Nome comum (NN)
atacou $\Longrightarrow$ Verbo no passado (VBD)
o $\Longrightarrow$ Determinante (DT)
gato $\Longrightarrow$ Nome comum (NN)

Figura 2.6: *Tagging* das palavras retiradas de um excerto de texto.

São ainda utilizados outros tipos de analisador sintáticos, baseados em gramáticas de contexto livre, que para além de identificarem o POS da palavra, conseguem identificar a função de uma palavra na frase, assim como as dependências entre palavras. A partir de uma gramática de contexto livre [5] é possível construir a sua árvore de sintaxe, como é demonstrado pelas figuras 2.7 e 2.8.

CFG

Sentence --> NP VP
VP --> Verb NP
NP --> O cão
NP --> o gato
Verb --> atacou

Figura 2.7: Representação da frase "O cão atacou o gato"numa gramática livre de contexto.

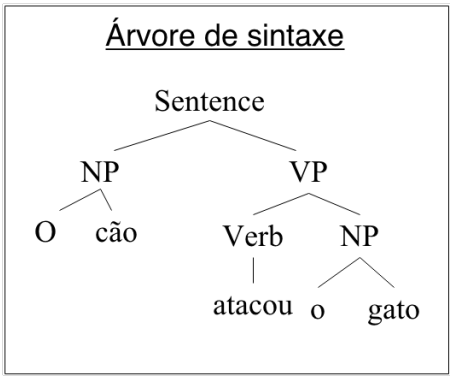


Figura 2.8: Representação da frase "O cão atacou o gato"numa árvore de *parsing*.

No que diz respeito à ambiguidade ao nível sintático, esta pode surgir no seguinte caso:

"Eu estou feliz por ser homem, assim como o João."

Este é um caso tradicional de ambiguidade sintática, em que a estrutura da frase pode assumir diferentes interpretações quando perguntamos porquê que o João está feliz, ou seja, ele pode estar feliz pelo sujeito da frase ser homem, assim como pode estar feliz por ser homem também.

2.1.3 Análise Semântica

A análise semântica é responsável por estudar o significado da linguagem. Para isso, é necessário proceder ao relacionamento entre a linguagem natural e a linguagem formal presente nos computadores, caso contrário não é possível a estes interpretarem sentidos de frases, palavras e textos.

Para a realização desse mapeamento entre linguagens, são conhecidas algumas técnicas de representação semântica em linguagem formal, como Predicados Lógicos [6], Redes Semânticas [7], *Frames* Semânticos [8]. Nas figuras 2.9, 2.10 e 2.11 é possível verificar essas representações.

"Coimbra é uma cidade que tem uma universidade."

- **Predicados Lógicos:**

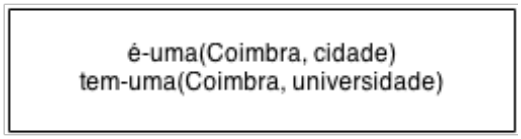


Figura 2.9: Representação através de lógica de predicados.

A lógica de predicados, também chamada de lógica de primeira ordem, passa por associar cada termo a um objeto que este representa. Relativamente a cada frase é associado um valor verdadeiro, permitindo assim retirar significado semântico para os termos da linguagem. A sintaxe simples e bem definida faz com que esta representação de conhecimento seja apropriada para os sistemas de informação.

- **Redes semânticas:**

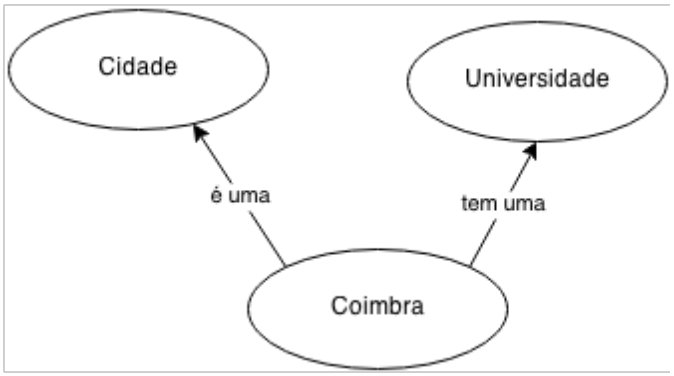


Figura 2.10: Representação através de uma rede semântica.

Uma rede semântica tem como objetivo representar conhecimento a partir da definição de um grafo direcionado, em que os nós representam conceitos e as arestas são as relações entre esses conceitos. Esta forma de representação permite ter uma melhor visibilidade sobre os objetos do domínio, devido à sua simplicidade e é flexível no que diz respeito à manipulação dos nós e arestas.

- ***Frames* Semânticos:**

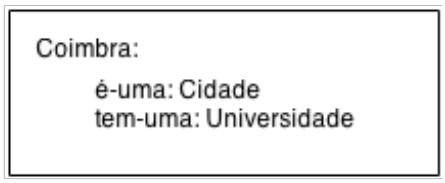


Figura 2.11: Representação através de um *frame* semântico.

Um *frame* é constituído por um nome, e procede à descrição de um objeto do domínio através de um conjunto de pares atributo-valor. A utilização de *frames* torna-se poderosa quando são introduzidas instruções nos seus atributos para processar informação contida em outros *frames*. Estes podem então ser facilmente implementados utilizando programação orientada a objetos.

A estrutura sintática pode por vezes ser utilizada como um *input* para a análise semântica, mas nem sempre esta é apropriada, por exemplo, numa estrutura sintática, por vezes, os elementos semanticamente mais relevantes estão muito dispersos uns dos outros, não sendo assim possível fazer um relacionamento entre eles. Outro caso é quando elementos da estrutura sintática não são nada relevantes para a análise semântica, mas que podem dar origem a resultados que não têm fundamento nenhum para o utilizador.

Relativamente ao tipo de ambiguidade que pode surgir neste tipo de análise, podemos dar como exemplo, um caso em que as palavras podem ter mais de um significado na frase:

- Eu hoje fui levantar dinheiro ao banco.
- Estive o jogo todo sentado no banco.

Na primeira frase, a palavra banco significa um local onde as pessoas armazenam o seu dinheiro, enquanto que na segunda, já nos estamos a referir a algo onde as pessoas se sentam. Nestes casos é necessário recorrer a uma técnica denominada por Word Sense Desambiguation (WSD) [9], que tem como objetivo selecionar o significado mais apropriado de uma palavra, de acordo com o contexto da frase.

2.2 Tecnologias da Web Semântica

Com a consequente evolução das tecnologias informáticas, a *web* tornou-se um elemento essencial no nosso quotidiano, isto porque a quantidade de informação que se encontra disponível *online* é cada vez mais essencial.

No entanto, devido ao volume de informação presente na *web*, os mecanismos de procura tradicionais, por si só não são suficientes, sendo necessário recorrer a outras técnicas que consigam representar a semântica. Para resolver esta situação, surgiu o conceito de Web Semântica (WS) por Tim Berners Lee [2], co-fundador e atual diretor do W3C (World Wide Web Consortium)[7], que defendia que os humanos e os computadores deviam trabalhar em conjunto. A WS passa a concentrar-se mais nos computadores, isto é, fazer com que estes executem tarefas de modo a agrupar informação mais relevante para o utilizador. No entanto, é necessário uma pessoa responsável por acompanhar as tarefas realizadas pelo computador, pois no caso de pesquisas na internet, os resultados destas são páginas *web* que estão preparadas para serem interpretadas por um humano e não por um computador.

Desta forma, foram criadas tecnologias que facilitam o relacionamento de informação na *web*, tornando assim o acesso e procura da mesma de uma forma mais precisa e mais completa no que diz respeito aos serviços de *Information Retrieval*. Algumas das normas estipuladas pelo W3C são por exemplo:

- ***Resource Description Framework (RDF)*** [10]: linguagem para representar informação presente na *web*, mais precisamente de páginas *web*.

[7] http://www.w3.org/

- ***RDF Schema*** [11]: utilizado para representar as propriedades e classes dos vários recursos RDF.
- ***Web Ontology Language (OWL)*** [12]: linguagem para descrever os conceitos e relações entre os metadados.
- ***SPARQL*** [13]: linguagem de interpretação de *queries* utilizada para recolher informação armazenada em formato RDF, no sentido de analisar o que o utilizador pretende consultar.

De modo a descrever as tecnologias da WS, Lee Feigenbaum utilizou a seguinte expressão [14]:

"*Semantic Web technologies are a set of technologies that happen to be especially well-suited for implementing semantic technology algorithms and solutions.*"

Basicamente, as tecnologias da WS são *standards* que se encaixam, perfeitamente, nas normas implementadas pelo W3C e assim simplificam a implementação das suas soluções. Casos disso são, a classificação dos dados tornar-se mais precisa devido a descrição de informação a partir dos *schemas* e das ontologias, ou então, a pesquisa semântica que necessita de dados representados conceptualmente (RDF) e de uma forma de pesquisar através desses conceitos (SPARQL).

Nas subsecções seguintes é feita a descrição de tecnologias que são utilizadas nesta dissertação.

2.2.1 Ontologias

No sentido de partilhar conhecimento entre as pessoas e os computadores, foi necessário desenvolver um mecanismo que estruturasse a informação de uma forma precisa. Para isso foram criadas as Ontologias, cujo o termo "Ontologia"remonta à filosofia e representa um ramo dedicado ao estudo da existência e da realidade [15].

No entanto, com o passar do tempo, o termo foi adotado no âmbito da computação, mais precisamente, nas áreas de Inteligência Artificial, Web Semântica, Representação de Conhecimento [16], Engenharia de Software [17], entre outras. Assim, foram criadas várias definições para o termo, sendo a mais citada na literatura da WS a de Grubber [18]:

"*An ontology is an explicit specification of a conceptualization*"

O que este pretende transmitir é que uma ontologia é uma visão abstrata e simples do mundo, em que os seus elementos estão claramente definidos e descritos por um vocabulário formal de modo a que estes sejam processados por um computador.

Uma outra definição conhecida e mais concisa foi elaborada pelo W3C [19] em que dizem o seguinte:

"OWL can be used to explicitly represent the meaning of terms in vocabularies and the relationships between those terms. This representation of terms and their interrelationships is called an ontology."

Aqui, o objetivo é referir que o uso de ontologias é uma mais valia quando se quer representar conhecimento e codificar o seu significado, através de um vocabulário estruturado. Assim, a partir desta definição é possível concluir que uma ontologia define um vocabulário, para uma determinada comunidade, que necessita de partilhar informação sobre uma determinada área.

Contudo, existem muitas definições para o termo "Ontologia", desde uma simples taxonomia até a uma descrição de conceitos e relacionamentos que devem ser considerados por um ou mais agentes. Esta diversidade de definições também depende da área onde esta está a ser utilizada, mas continua a ser persistente.

Classificação de ontologias

As ontologias podem assumir diferentes categorias, dependendo do seu ponto de vista. Assim, estas podem ser classificadas relativamente ao seu nível de generalidade, sendo demonstrado de seguida a proposta feita por Guarino [20]:

- ***Top-level Ontologies:*** Aqui são estruturados conceitos muito gerais que são independentes de qualquer domínio, mas que podem dar origem a vários (ex. espaço, tempo, evento, ação).
- ***Domain Ontologies:*** Este tipo de ontologias descrevem vocabulário que está relacionado com um determinado domínio (ex. Medicina, Automóveis).
- ***Task Ontologies:*** Aqui o processo é semelhante às de domínio, mas são associadas a uma tarefa ou atividade executada (ex. Comprar, Estagiar).
- ***Application Ontologies:*** Apropriada para descrever conceitos que sejam dependentes de um determinado domínio e tarefa e que são, normalmente, especificações das ontologias relacionadas. Em maior parte dos casos, estes conceitos correspondem a papéis interpretados por entidades de um domínio enquanto executam uma atividade ou tarefa.

Ou podem ser classificadas com base na sua estrutura de conceptualização [21]:

- ***Terminological Ontologies:*** São especificados os termos que posteriormente são utilizados para representar o conhecimento no domínio do discurso.

- ***Information Ontologies:*** Neste tipo é especificada a estrutura de armazenamento das base de dados. O armazenamento de informação é semelhante a uma base de dados.
- ***Knowledge Modeling Ontologies:*** Especificam conceptualizações do conhecimento, têm uma estrutura interna mais rica do que as *Information Ontologies* e estão otimizadas para a utilização do conhecimento que descrevem. Segundo Heijst, este tipo de ontologias são as mais interessante, dentro do contexto de *Knowledge Base Systems*.

Processo de Construção

Na fase de construção de uma ontologia existem diversos tipo de processamento. No entanto, existe uma metodologia que é muito utilizada e que foi desenvolvida por Noy e McGuinness [22]. Esta é uma abordagem iterativa que segue os seguintes passos de desenvolvimento:

1. **Identificar o domínio e o âmbito da ontologia:** Neste primeiro passo, pretende-se definir os objetivos da ontologia, assim como o tipo de aplicações em que esta pode ser utilizada. Relativamente ao propósito este pode ser definido através de questões em linguagem natural que descrevem o que a ontologia deve responder. No entanto, estes elementos podem ser todos alterados durante o processo de *design*, mas facilitam, em muito, a definição do âmbito do modelo de ontologia.
2. **Reutilizar ontologias existentes:** A partir da utilização de ontologias já existentes, é possível reduzir a carga de trabalho, visto que em maior parte dos casos apenas é necessário aperfeiçoar um elemento ou outro, de modo a adaptar ao domínio em que estamos a trabalhar.
3. **Definir os termos importantes da ontologia:** É importante definir previamente os termos relacionados com o domínio da ontologia, no sentido de conseguir perceber quais são as instâncias necessárias, assim como as propriedades das mesmas. No entanto, não é preciso pensar nas relações entre elas.
4. **Definir as classes e as suas hierarquias:** A definição das classes e hierarquias é bastante importante, pois se estiverem bem estruturadas no inicio do processo evitam-se problemas de grande dimensão.

   Para estruturar a hierarquia das classes existem várias abordagens, sendo que as mais comuns são as seguintes:

   (a) ***Top-down:*** Nesta abordagem, começamos por definir os conceitos mais gerais e por fim os mais específicos.

(b) ***Bottom-up:*** Aqui são definidos primeiramente as classes mais específicas e de seguida estas são associadas a classes mais gerais.

(c) ***Combination of both:*** Desta forma, estruturamos primeiro os conceitos mais importantes para a ontologia, sejam eles mais gerais ou mais específicos.

5. **Definir as propriedades das classes:** Após a criação das classes, estas vão ter propriedades associadas tendo assim de ser definidas. Por exemplo, no caso de uma classe chamada *"SMS"*, podemos associar uma propriedade *"date"*.

6. **Definir as restrições das propriedades:** Para cada uma das propriedades devem ser criadas restrições, isto é, as propriedades podem ter de respeitar um determinado tipo (*String, Number, Date, etc*), podem variar quanto à sua cardinalidade ou então podem ser feitas restrições quanto aos valores que são permitidos para essa propriedade.

7. **Criar as instâncias:** A criação de instâncias é o ponto final deste processo. Aqui temos de proceder à criação de uma instância de uma determinada classe e, de seguida, preencher devidamente os valores das suas propriedades.

As ontologias são um fator muito importante para a WS, tendo de estar devidamente estruturadas, pois são responsáveis por representar o conhecimento de um determinado domínio. No entanto, existem outros tópicos que devem ser referidos durante esta dissertação que são de igual importância para o bom funcionamento da WS.

2.2.2 Dados Semânticos

No ambiente da Web Semântica, é necessário ter recursos, no que diz respeito à informação, de modo a que os vários processos dentro desta área consigam melhorar os seus resultados face aos seus objetivos. Em termos semânticos estes recursos são denominados como dados ou dados semânticos. Estes dados assumem o papel de descrever os vários objetos como entidades e relações. O tipo de dados semânticos mais usado, por exemplo, no caso de pesquisas semânticas são os RDF. Estes são estruturados como um grafo que representam as suas entidades, o seus atributos e as suas relações como um conjunto de triplos. Para demonstrar mais detalhadamente a sua representação, Tran e Mika [23] apresentam um modelo de grafo que captura informação sobre dois investigadores (ver figura 2.12).

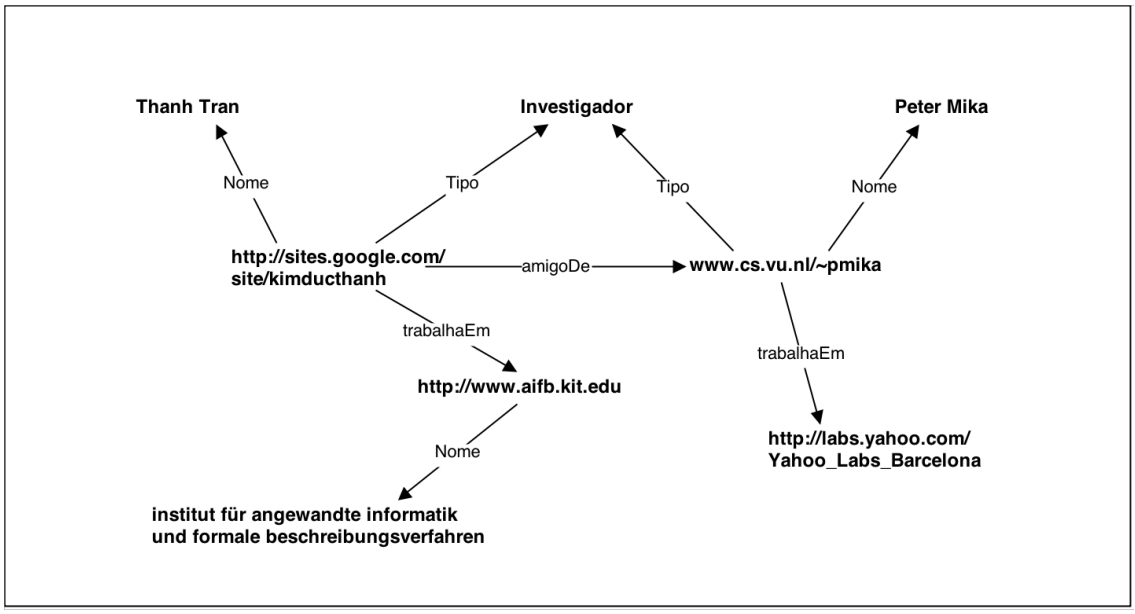


Figura 2.12: Representação da informação de dois investigadores através de um grafo.

Em maior parte dos casos, este modelo de grafo é construído a partir de elementos XML[8] ou bases de dados relacionais, sendo que os diferentes tipos de informação possíveis de estruturar estão públicos na *web*, assim como incorporados em vários mecanismos de pesquisa. Esta informação, após ser convertida para RDF's passa a ser chamada de dados semânticos. Assim, Tran e Mika, atribuem a seguinte definição para o termo:

"Semantic data can be conceived as a graph, where nodes represent entities and their attribute values, and edges stand for attributes of entities or relations between entities."

Nos dias de hoje vários sistemas já são desenvolvidos com o intuito de concentrarem os seus esforços, na obtenção de dados semânticos, enquanto que por exemplo, os sistemas de pesquisa tradicionais baseiam-se em dados brutos[9]. Nestes a informação é extraída a partir de objetos como áudio, vídeo, imagens e textos, não estando estes tratados devidamente, isto é, não existe uma obtenção de informação a partir de entidades e relações. No entanto, existem alternativas para conseguir capturar esta informação a partir de elementos semânticos, tendo estes de sofrer um processo de reconhecimento de entidades e relações. A utilização da técnica Reconhecimento de Entidades Mencionadas (REM) é uma solução para obter uma representação semântica de *raw data*, sendo esta traduzida pelos termos metadados semânticos ou anotações. Tran e Mika, referem-se a metadados semânticos como qualquer tipo de dados semânticos que oferecem informação adicional acerca dos objetos a serem procurados. O termo anotação ou a utilização de RDFa[10] está associada aos metadados de uma página *web* e tem como objetivo descrever, de

[8] http://www.w3.org/standards/xml/
[9] *Raw data* - http://en.wikipedia.org/wiki/Raw_data
[10] http://www.w3.org/TR/xhtml-rdfa-primer/

uma forma estruturada, o texto contido nessa página.

No entanto, os dados semânticos apenas interpretam entidades concretas ou instâncias, isto é, são definidos valores de propriedades e de relação entre entidades. Como tal, existem outras abordagens para os mecanismos de procura que serão analisados nos próximos tópicos.

2.2.3 Modelos Semânticos

Como foi referenciado, existem outras fontes de semântica, denominados por modelos semânticos. Estes são divididos em diferentes categorias que passamos a analisar.

Modelos de Conhecimento

Este tipo de modelos visa representar conhecimento a partir de três elementos principais: classes, atributos e relações, ou seja, representar classes das entidades, atributos que pertencem a essas classes e as relações entre as mesmas. É possível visualizar no seguinte esquema, proposto por Tran e Mika, como estes elementos se relacionam (ver figura 2.13).

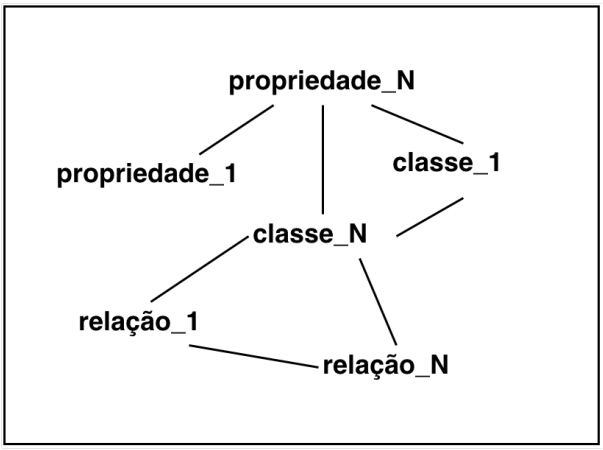


Figura 2.13: Classes, atributos e relações representados num modelo de conhecimento.

Os modelos de conhecimento tornam-se então responsáveis por armazenar os vários tipos de informação que posteriormente servem de suporte à pesquisa semântica, sendo em maior parte dos casos este modelo de conhecimento estruturado como uma ontologia.

A utilização destes modelos providencia uma navegação rápida entre a informação armazenada. Este ponto torna-se crucial quando estamos a falar de sistemas de pesquisa avançados, com grandes volumes de informação. Para além disso, ao integrar modelos de conhecimento na *query* e na navegação, é possível obter informação de acordo com o contexto pretendido.

Modelos Lexicais

Ao contrário dos modelos de conhecimento, este tipo de modelos passam a recolher elementos semânticos ao nível das palavras, ou seja, os seus conceitos correspondem ao sentido das palavras. Um exemplo de modelo lexical que é muito utilizado é um *thesaurus*[11], que pode assumir a forma de um dicionário ou lista. Este tem a função de agrupar as diferentes palavras de acordo com a sua similaridade semântica, sendo que cada um destes conjuntos representa um sentido de palavra diferente. No entanto, também são tidas em conta outras relações para este agrupamento como por exemplo, sinónimos, relações pai/filho, entre outras. O *WordNet* [24] é um bom exemplo de modelo lexical, em que não se concentra apenas no sentido e nas variantes lexicais da palavra, mas também as relaciona através dos seus sinónimos.

No esquema proposto por Tran e Mika (figura 2.14) é possível observar um esquema de conceitos relacionados.

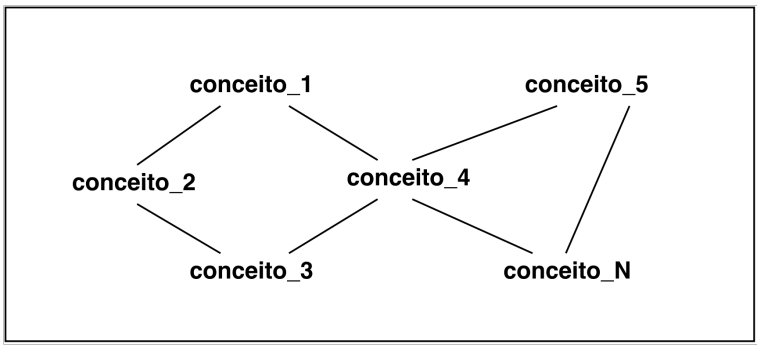


Figura 2.14: Relação entre os vários conceitos num modelo lexical.

De forma a resumir a função dos modelos lexicais no âmbito da representação semântica, podemos dizer que estes tem como objetivo organizar os diferentes conceitos das palavras de acordo com as relações lexicais que são possíveis detetar entre as mesmas.

Estes dois tópicos, dados semânticos e modelos semânticos, permitem-nos representar as diferentes maneiras de como os elementos semânticos são obtidos nos diferentes mecanismos de pesquisa semântica.

2.2.4 Pesquisa Semântica

A pesquisa semântica surge no sentido de melhorar o processo de pesquisa do utilizador nos diferentes sistemas que este utiliza. Aqui pretende-se atingir uma maior precisão de resultados, sendo estes baseados na intenção que o próprio utilizador tem, no contexto em que os termos

[11] http://en.wikipedia.org/wiki/Thesaurus/

introduzidos na pesquisa se inserem, entre outros. Nos sistemas de pesquisa tradicionais é preciso ter em conta os seguintes elementos, a *data*, a *query* e uma *framework* que nos permita processar os dados resultantes de uma *query*. No entanto, apenas com estes componentes, os resultados podem não ser satisfatórios, pois a *query* introduzida pode dar origem a ambiguidades. Este é um dos principais problemas que a pesquisa semântica pretende solucionar, ou seja, utilizar semântica no sentido de conseguir melhorar a experiência de procura.

Tran e Mika, apresentam o seguinte esquema (figura 2.15) para representar como a semântica influencia a interpretação dos dados e da *query*, assim como também é relevante para a apresentação de resultados.

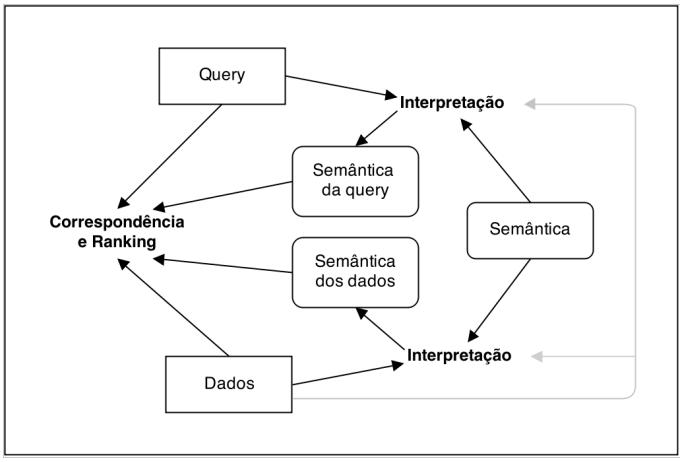


Figura 2.15: Representação do processo de pesquisa semântica.

Podemos verificar então que os elementos semânticos são processados, tanto para a *query* introduzida como para a *data* fornecida. Estes elementos semânticos abordam tópicos referidos anteriormente, como modelos semânticos que em conjunto com os componentes fornecidos pela *data* (dados semânticos, metadados semânticos, etc), vai detetar conceitos, entidades e relações na informação providenciada, dando origem a elementos semânticos dos dados. Por outro lado, quando existe uma interpretação a partir dos elementos da *query* (palavras-chave, linguagem natural, etc) com modelos semânticos, vamos obter modelos semânticos sobre a *query*.

De acordo com o esquema apresentado, o processo de pesquisa semântica pode ser descrito da seguinte forma:

1. Os recursos semânticos (à direita na figura 2.15) que se baseiam em modelos de conhecimento e lexicais, são processados e interpretados quer com a *query*, quer com os dados.

2. Relativamente à interpretação a partir da *query* (parte superior da figura 2.15), esta processa palavras-chave e linguagem natural provenientes da *query*, modelos semânticos e dados semânticos (estes últimos representados na linha a cinzento), no sentido de detetar

conceitos e, fazer a tradução de palavras-chave e *queries* de linguagem natural. A partir deste processo são gerados elementos semânticos da *query* (conceitos, entidades, relações).

3. No que diz respeito à interpretação dos dados (parte inferior da figura 2.15) o processo é semelhante, no entanto aqui para além dos dados semânticos, metadados e modelos semânticos, são interpretados também dados brutos (*raw data*) que são utilizados para deteção de conceitos, entidades e relações, dando origem aos elementos semânticos dos dados.

4. Como tal, estes dois tipos de recursos semânticos vão sofrer vários processos (correspondência de termos, travessia de grafos, etc) no sentido de gerar resultados que posteriormente vão auxiliar no processo de classificação (*ranking*).

A pesquisa semântica torna-se então valiosa para utilizador, visto que a partir da mesma, este vai conseguir obter um melhor conjunto de resultados, que se encontram mais adequados ao que este pretende encontrar.

2.3 Definição de Contexto

Baseado no *Merriam-Webster's Collegiate Dictionary*[12], a palavra contexto pode ser definida por:

"The interrelated conditions in which something exists or occurs"

No entanto, a palavra pode ser usada em diferentes áreas, porque qualquer ação que seja feita é sempre baseada num contexto. Este encontra-se sempre associado a uma entidade, é utilizado para resolver ambiguidades, varia com a localização das pessoas, entre outros. Na área da computação, o termo contexto assume diferentes áreas, sendo que para esta dissertação vai ser focado o contexto extraído de ambientes *mobile*.

Relativamente à sua modelação, esta pode ser de diferentes categorias segundo Zimmerman [25]. Este classifica o contexto a partir de cinco categorias que se tornam fundamentais quando estamos a tratar de sistemas baseados em contexto:

- **Contexto individual:** Este tipo de contexto contém informação relacionada com tudo o que pode ser observado acerca de uma determinada entidade.

[12] http://www.merriam-webster.com/

- **Contexto baseado no tempo:** A informação presente neste tipo de contexto está relacionada através do tempo. Desta forma é possível tirar partido de hábitos de uma entidade para prever futuros acontecimentos.
- **Contexto baseado em localizações:** Com a crescente utilização de dispositivos móveis este tipo de contexto tornou-se cada vez mais essencial, visto que oferece uma grande quantidade de informação sobre uma entidade.
- **Contexto baseado em atividades:** Armazena informação sobre atividades praticadas pelo utilizador, demonstrando de certa forma as necessidades do mesmo.
- **Contexto baseado em relações:** Para este tipo de contexto, são armazenadas relações entre várias entidades, podendo ser estas pessoas, dispositivos, entre outros.

Com isto, é perceptível a importância que a inclusão de contexto transmite, e que desta forma é possível, num contexto de computação móvel extrair imenso conhecimento no sentido de desenvolver sistemas que vão de encontro às necessidades do seu utilizador.

Estando a computação móvel, assim como a utilização de *smartphones* por parte da sociedade num sentido crescente, é perceptível que a quantidade de informação sobre o utilizador armazenada no dispositivo móvel é imensa. A informação que pode ser extraída abrange diversas categorias, como por exemplo, *emails*, mensagens, contactos, localizações, informação das redes sociais que o utilizador se encontra, entre outros. A partir desta informação é possível ter uma maior aproximação ao contexto do utilizador como por exemplo, assuntos pelos quais este se encontra mais interessado, locais que este gosta ou costuma frequentar e com que tipo de indivíduos costuma comunicar. Esta proximidade deve-se ao facto de o *smartphone* acabar a ser um dispositivo muito próximo do utilizador, funcionando como uma extensão do seu corpo no que diz respeito ao armazenamento de informação.

2.4 *Information Retrieval*

O conceito de *Information Retrieval* (IR) [3], surgiu no momento em que se procurava utilizar os computadores para procurarem informação relevante para o utilizador de uma forma autónoma. Assim, foi necessário implementar mecanismos que permitissem reduzir o *overload* de informação que era apresentado no ato de uma pesquisa. Atualmente, a área de IR está em crescimento e cada vez é mais é procurada a sua utilização. No entanto, os grandes mecanismos de pesquisa *web* já se encontram a funcionar com estas técnicas, como por exemplo o *Google*[13], *Bing*[14], entre

[13]http://www.google.com/
[14]http://www.bing.com/

outros. Um processo de IR, de uma forma geral, é representado pela figura 2.16.

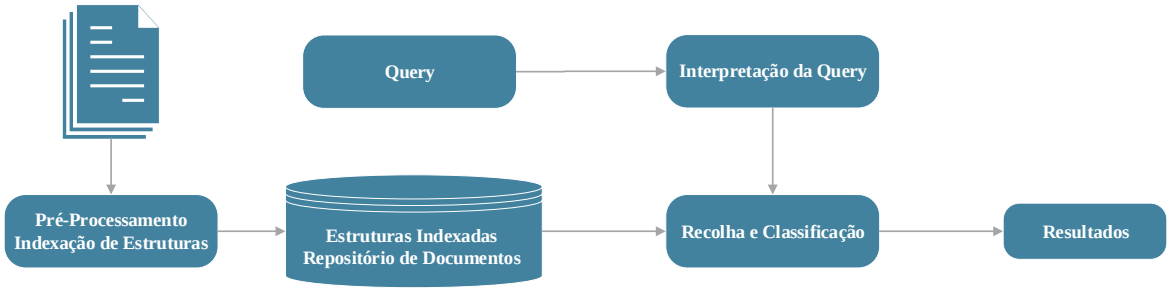


Figura 2.16: Representação do processo de *Information Retrieval*.

E pelos seguintes passos:

- O utilizador expressa a informação que necessita, normalmente, através de uma *query*, sendo esta composta por um ou vários conceitos em questão.
- São feitas operações de pré-processamento aos documentos de texto e uma análise à *query* no sentido de apenas ficarem os termos relevantes.
- Após a interpretação da *query* e a indexação das estruturas, é feito um processo que recolhe os documentos que mais se adequam à *query* para que estes sejam classificados de acordo com o seu grau de correspondência à mesma.
- Os documentos são apresentados ao utilizador de acordo com o seu grau de correspondência perante a *query*.

Nas seguintes subsecções vamos fazer uma abordagem do processo de uma forma mais interna, isto é, como se encontram representados os documentos, que tipo de operações de pré-processamento são feitas para a *query* e para os documentos, assim como posteriormente estes são indexados. Para além disso, são abordados os diferentes tipos de *queries* e como estas são interpretadas, assim como são apresentados os mecanismos mais relevantes para a classificação e apresentação de documentos.

Por fim, e de acordo com o que vai ser desenvolvido para esta dissertação, vamos abordar a forma como o contexto é combinado com os processos de IR, e como é que este influencia positivamente os resultados obtidos.

2.4.1 Representação do Documento

Existem diversas maneiras de fazer a representação dos documentos, sendo estas baseadas em modelos que pretendem descrever os documentos como um conjunto de termos. Esses termos vão ter pesos associados que definem a relevância do mesmo em relação ao documento. Aqui vamos apresentar duas representações que são muito utilizadas para os processos de IR.

- **Modelo Booleano (MB):** Neste tipo de representação, a relevância dos termos é feita a partir de uma classificação binária dos mesmos. Assim, o peso dos termos assume o valor 0 ou 1, sendo que "0"significa que o termo não está presente no documento e "1"significa precisamente o contrário. As *queries* quando são interpretadas neste modelo, são assumidas como um conjunto de termos que se encontram ligados através de operadores lógicos (*and, not*). Para finalizar este processo, o documento vai ser considerado relevante para a *query* se os seus termos estiverem de acordo com as restrições lógicas impostas. É apresentado um exemplo na figura 2.17.

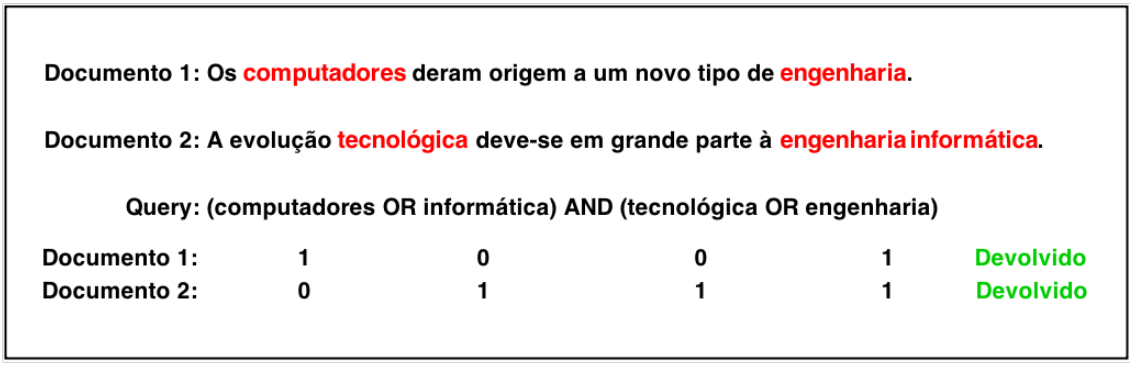


Figura 2.17: Identificação de termos relevantes para a *query* através do modelo booleano.

- **Modelo Vetorial (MV):** Este modelo é, atualmente, o mais utilizado pelos sistemas de IR. A razão para tal é este modelo ser considerado muito eficiente, assim como muito simples de implementar. O MV foi proposto por *Salton* [26], com o objetivo de suprimir as limitações presentes no MB, isto é, utilizar uma atribuição de pesos aos termos de uma forma não binária, e também conseguir fazer correspondências parciais entre os documentos e as *queries*.

  A partir deste novo modelo, é então possível apresentar uma lista de documentos que estão classificados de acordo com o seu grau de similaridade relativamente à *query*. Este grau é calculado a partir da correlação dos dois vetores (*query* e documento). Esta solução torna-se benéfica pelo facto de ser possível obter resultados a partir da aproximação dos termos do documento aos termos da *query*, mas também pode apresentar os seus problemas pelo facto de assumirmos que os termos são independentes entre si.

Existem diversas maneiras para calcular os pesos dos termos, sendo o processo mais comum a técnica *Term Frequency/Inverse Document Frequency* (TFxIDF). Este processo divide-se em três fases distintas. A frequência de um termo num determinado documento é representado pelo seguinte cálculo:

***Term Frequency* (TF)**

$$tf_i = \frac{t_i}{\sum_{i=1}^{k} t_k} \tag{2.1}$$

Em que tf_i representa a frequência do termo i no documento, t_i indica o número de vezes que o termo i aparece no documento e k representa o número de termos no documento.

A segunda equação permite-nos calcular se um determinado termo tem uma presença comum ou rara no conjunto de documentos. É definida pela seguinte expressão:

***Inverse Document Frequency* (IDF)**

$$idf_i = \log\left(\frac{N}{n_i}\right) \tag{2.2}$$

Aqui, idf_i representa a frequência inversa do termo i no documento, N indica o total de documentos e n_i o número de documentos em que o termo i aparece.

Para obter o peso associado a um termo, num determinado documento procede-se ao produto entre o TF e o IDF:

***Term Frequency/Inverse Document Frequency* (TFxIDF)**

$$TF/IDF_i = \frac{t_i}{\sum_{i=1}^{k} t_k} \times \log\left(\frac{N}{n_i}\right) \tag{2.3}$$

Assumindo os cálculos já feitos a partir das fórmulas anteriores, é apresentado um exemplo desta representação (ver figura 2.18).

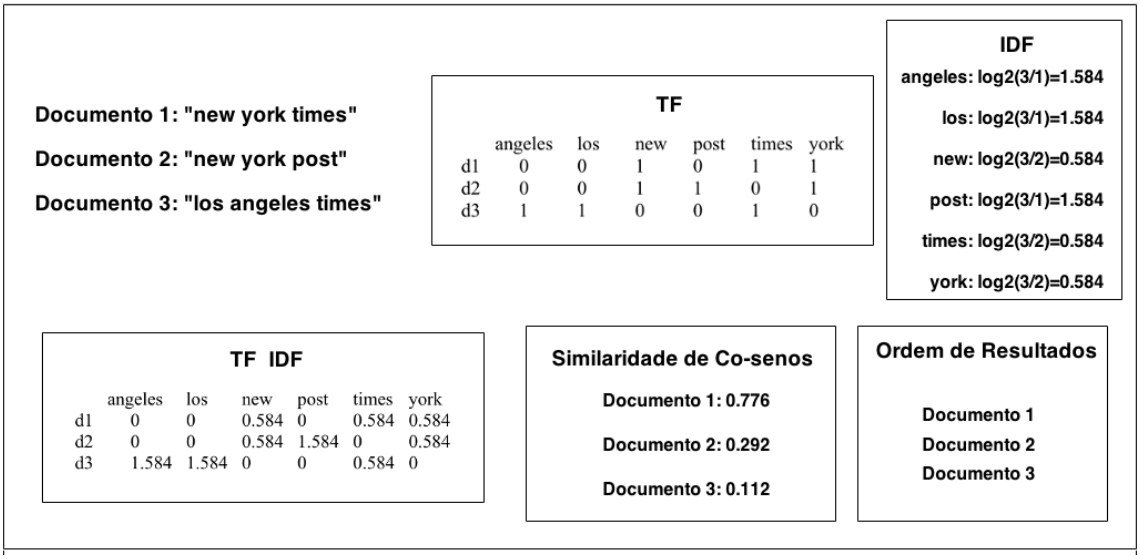


Figura 2.18: Representação de documentos num modelo vetorial e a sua ordenação face à *query*.

Por fim, ficam apresentadas as vantagens e desvantagens (ver tabela 2.2) dos dois modelos referidos, sendo que o modelo vetorial é o que apresenta melhores garantias.

	Vantagens	**Desvantagens**
Modelo Booleano	- Simplicidade	- Relevante ou não relevante (0 ou 1) - Não suporta resultados parciais
Modelo Vetorial	- Peso dos termos melhora a recolha - Suporta resultados parciais - Documentos ordenados por *ranking* - Simples e rápido	- Termos independentes entre si

Tabela 2.2: Vantagens e desvantagens do *Modelo Booleano e Modelo Vetorial*

2.4.2 Análise de Documentos

No sentido de conseguirmos um processamento de informação, são necessárias medidas que tornem essa informação menos vasta. Na área de *Information Retrieval* quando estamos a proceder à representação de um documento é comum fazer uma análise prévia, um pré-processamento do texto no sentido de fazer uma selecção dos termos que vão ser indexados.

Pré-Processamento

O pré-processamento do texto, segue um conjunto de operações que permitem reduzir o número de termos a serem indexados, assim como permitem ajustar alguns à sua forma original de modo a evitar repetição de conceitos. O seguinte esquema visa representar as várias operações que podem ser feitas neste sentido (ver figura 2.19).

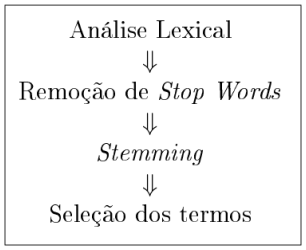


Figura 2.19: Esquema representativo das várias etapas do pré-processamento.

A **análise lexical**, pretende fazer a divisão do texto em várias frases ou em vários *tokens (Tokenization*, a partir desta divisão é possível fazer uma análise mais detalhada a cada termo. No entanto, é preciso ter em atenção esta análise, pois podem surgir alguns problemas. Para esta dissertação, o facto de extrairmos contexto das SMS's, pode ser um grande problema, como por exemplo nos casos já referidos na subsecção 2.1.1: abreviações de palavras, frases sem pontuação, *emoticons*, etc.

A **Remoção de *Stop Words*** e a técnica de ***stemming***, visam então tornar as várias palavras presentes no documento, mais específicas, reduzindo estas a formas normalizadas, assim como retirar palavras que não vão ter nenhuma utilidade para o processo de IR.

Após estes passos estarem concretizados, o nosso documento fica reduzido a um conjunto de termos que são candidatos a serem indexados. Para proceder à **seleção dos termos** que realmente vão ser indexados, são utilizadas diversas abordagens. Existem alguns casos em que os termos selecionados são apenas nomes, pois estes podem conter muita informação semântica do documento. No entanto, outra técnica utilizada é a seleção dos termos mais frequentes, mas nem sempre os termos mais frequentes são os mais descritivos.

Indexação de Estruturas

No contexto de IR, o conceito mais utilizado para a indexação de estruturas é denominado por *Inverted Files*. Este tipo de estrutura permite fazer procuras de uma forma rápida, ao contrário de outras estruturas como é o caso das procuras sequenciais, que tem uma mau desempenho quando estamos a tratar uma grande quantidade de informação. Este pode estar estruturado

ao nível da palavra, ao nível da frase ou ao nível do caracter. São utilizados dicionários para o processo, pois estes oferecem uma hierarquia que permite adaptar a *query* às necessidades do utilizador.

A escolha do conceito de *Inverted Files*, tem um bom *trade-off*, isto porque na utilização de *suffix Trees* é possível ter uma complexidade temporal muito baixa, mas no entanto a complexidade espacial é muita elevada, pois há uma necessidade de indexação extrema. No entanto, quando fazemos uso de *Inverted Files* esses dois tipos de complexidade são relativamente baixos, acabando por se compensarem um ao outro.

2.4.3 Interpretação da Query

Para apresentar ao utilizador a informação que este necessita, é necessário proceder a uma interpretação do que o mesmo escreveu, ou seja, interpretar/estruturar a *query*. Esta pode assumir diferentes categorias dependendo do tipo de sistema que foi implementado. As três categorias que mais se distinguem são apresentadas por Baeza-Yates e Ribeiro-Neto [27], sendo que dentro destas categorias elas assumem diferentes tipos.

- **Baseada em palavras-chave:** Este tipo de *queries* são combinações das várias palavras que satisfazem a necessidade do utilizador. Podem ser do tipo ***single word***, em que a *query* é formulada apenas por uma palavra, sendo então os documentos que contém este termo devolvidos e ordenados de acordo com a frequência do termo. As ***queries* de contexto**, tratam de procurar a partir de um determinado contexto, podendo estas correspondências ser em forma de frase, ou então por proximidade. Na primeira é feita uma correspondência entre as palavras presentes na *query* e o documento de uma forma sequencial. Por proximidade, o processo é semelhante mas é estipulado um distância máxima entre a palavra/sequências de palavras presentes na *query* e o documento. Por outro lado, as ***boolean queries*** descrevem a informação relacionando as várias palavras através de operadores lógicos (*and, or*), o que determina que os documentos a serem apresentados têm de respeitar as restrições feitas por esses operadores. Por fim, as ***natural language queries***, são consideradas com enumerações de palavras e de *queries* de contexto, que não necessitam explicitamente de operações lógicas. No entanto, é necessário ter alguma "inteligência"no processo o que faz com que o uso destas ainda seja um desafio.

- **Correspondência por padrões:** Neste tipo de *queries*, são utilizados padrões para fazer a correspondência com as palavras. Para uma palavra corresponder a um padrão, basta a mesma ser igual a uma das palavras definidas pelo padrão. Um padrão é então um conjunto de critérios sintáticos que ocorrem num determinado segmento de texto, que podem variar

na sua complexidade, isto é, podem tratar-se apenas de um conjunto de palavras, podem ser expressões regulares ou até mesmo de similaridade.

- ***Queries* estruturais:** Este tipo de *queries* permitem fazer operações mais poderosas, isto é, a partir de documentos que se encontram bem estruturados é possível utilizar *queries* que se baseiam na estrutura do mesmo. São assumidas três estruturas como principais.

  1. **Estrutura Fixa:** Neste tipo de estruturas, o documento é dividido em vários campos, como se fosse um formulário. Um bom caso de uso é quando estamos a falar de *emails*, em que temos os vários campos do mesmo (Destinatário, Assunto, etc).
  2. **Estrutura Hipertextual:** Aqui o tipo de estrutura assenta num grafo direcionado, em que cada um dos nós contém um bocado de texto, podendo cada um dos nós representar uma secção, um parágrafo ou um documento inteiro.
  3. **Estrutura Hierárquica:** Para esta são utilizados modelos como árvores que permitem decomposição recursiva dos documentos.

Existem outros processos que visam melhorar a interpretação da *query*, isto porque por vezes é difícil para o próprio utilizador conseguir formular *queries* que lhe garantam que os resultados devolvidos, vão ao encontro do que este pretende. Esses processos são chamados de métodos locais e métodos globais. Os métodos locais, concentram-se em ajustar a *query* relativamente ao conjunto de resultados inicialmente apresentados para esta. O método mais comum é denominado por *relevance feedback*, que recolhe *feedback* do utilizador para reformular a *query* de uma forma iterativa. Os passos para este processo são descritos de seguida:

- o utilizador formula uma *query*
- o sistema devolve um conjunto de resultados iniciais
- o utilizador faz a classificação manual
- o sistema processa uma melhor representação dos documentos relevantes, baseados no *feedback* do utilizador
- o sistema apresenta o novo conjunto de resultados

Relativamente aos métodos globais, o mais utilizado é chamado por *query expansion*. Este processo tem como objetivo expandir ou reformular a *query* com base no conjunto total de documentos. Podem ser utilizadas as seguintes técnicas de forma a expandir a *query*:

- encontrar sinónimos para as palavras presentes na *query*, e procurar por estes (utilização de *thesaurus*)

- aplicar a técnica de *stemming* a cada palavra
- correção automática de termos presentes na *query* e respetiva pesquisa

2.4.4 Recolha e Ordenação de Resultados

As questões que sobressaem agora são de como, um processo de IR apresenta e classifica os resultados, isto é, como é que estes processos são executados de modo a que seja possível o utilizador ficar satisfeito com os resultados que lhe são apresentados. Para isso, são conhecidas algumas abordagens, como por exemplo o já referido **TFxIDF**, a técnica de **similaridade de co-senos** [28] ou em relação a páginas *web* temos um caso muito conhecido como o *Page Rank*[15] da *Google*. Este processo classifica uma página de acordo com as referências que são feitas sobre ela a partir de outras páginas.

No entanto, este processo por si só não é suficiente, sendo muitas vezes necessário fazer ajustamentos a uma lista que se encontra com um *ranking* atribuído. Com este processo, é possível aumentar em muito a qualidade dessa lista de documentos e ir ao encontro da informação que o utilizador pretende. Para esse ajustamento, são conhecidas diversas abordagens. As mais comuns no contexto de IR, são denominadas por abrangência e precisão.

Abrangência: No contexto de IR, esta medida é vista como a fração de documentos relevantes para a *query* que foram apresentados ao utilizador. Por outras palavras, mede os documentos relevantes que foram devolvidos. De seguida é apresentada a fórmula para este processo:

$$abrangencia = \frac{DocumentosRelevantes \cap DocumentosRecolhidos}{DocumentosRelevantes} \tag{2.4}$$

Precisão: Por outro lado, quando se fala de precisão, o seu objetivo já passa por medir a partir dos documentos que foram apresentados nos resultados, quantos é que eram relevantes para a *query*. No contexto de documentos, precisão é o número de resultados corretos divido pelo número de resultados apresentados. Por vezes, esta medição pode apenas ser feita ao conjunto de alguns dos resultados devolvidos.

$$precisao = \frac{DocumentosRelevantes \cap DocumentosRecolhidos}{DocumentosRecolhidos} \tag{2.5}$$

Contudo, foi proposta uma técnica que pretende combinar abrangência e precisão numa única medida. Esta é utilizada para conseguir um melhor balanceamento entre as duas medidas, pois

[15] http://en.wikipedia.org/wiki/PageRank

ao utilizar este processo, pretende-se obter um *ranking* final com uma abrangência e precisão altas.

$$f - measure = 2 \times \frac{Abrangencia \times Precisao}{Abrangencia + Precisao} \tag{2.6}$$

2.4.5 *Context-based Information Retrieval*

Indo de encontro ao foco principal desta dissertação, surge então a questão do uso de contexto no sentido de melhorar o processo de IR. Devido ao crescimento nos processos de IR, são cada vez mais investigadas formas de aprimorar os resultados, sendo que a introdução de contexto foi afirmada como uma solução efetiva [29]. Existem cada vez mais formas de obter contexto do utilizador, isto porque os desenvolvimentos tecnológicos estão cada vez mais avançados, principalmente no que diz respeito aos dispositivos móveis. A quantidade de contexto que é possível recolher sobre um utilizador é enorme, isto porque cada vez mais há um contacto entre as pessoas e um dispositivo, quer seja este um computador, um *smartphone* ou um *tablet*. No entanto, a implementação de sistemas de IR com contexto de utilizador ainda é um grande desafio, e continuam a ser opção sistemas que se baseiam apenas na *query* e nos documentos, abstraindo-se assim de qualquer outro tipo de informação como os interesses do próprio utilizador. No entanto, para esta dissertação, vai ser feita uma abordagem de como a partir de um simples *smartphone*, o utilizador se torna uma fonte rica de contexto para ser introduzido nos processos de *Information Retrieval*.

2.5 Trabalho Relacionado

Nesta secção são apresentados trabalhos realizados que, de certa forma, conseguiram obter resultados com a introdução de contexto proveniente do utilizador. Desta maneira é possível observar que influência já tem a utilização de contexto nos diferentes tipos de aplicações.

2.5.1 Pesquisa Contextual Utilizando Perfis de Utilizadores Baseados em Ontologias

Deparados com o facto de os mecanismos de procura apresentarem resultados ao utilizador sem qualquer tipo de comparação com os interesses do mesmo, Challam [30], decidiram construir perfis de utilizadores baseados nas várias operações que o utilizador estivesse a fazer durante o ato de pesquisa.

O processo passava por uma aplicação *Windows*[16] que se encontrava constantemente a monitorizar o que o utilizador estava a fazer, páginas *web* abertas, aplicações do *Microsoft Office*[17], ou mesmo documentos do *MSN Messenger*[18].

Quando o utilizador fazia a pesquisa, o contexto até ao momento armazenado era utilizado para criar o perfil contextual do mesmo através de uma ontologia. A *query* era interpretada pelo sistema e apresentava os dez resultados com melhor *ranking*. Posteriormente, estes resultados eram re-ordenados a partir da combinação do *ranking* original e da sua similaridade com o perfil contextual do utilizador. Foi utilizada a técnica de similaridade de co-senos para fazer esta combinação, sendo que os pesos dos conceitos presentes foram calculados a partir da técnica *Term Frequency/Inverse Document Frequency*.

Para avaliarem o impacto dos perfis contextuais do utilizador nos resultados obtidos da *web*, foi criado um *wrapper* em torno do mecanismo de pesquisa da *Google*, a partir da API dos mesmos. Esse *wrapper* criava um registo das *queries* introduzidas pelo utilizador, os respetivos resultados devolvidos, o resultado que o utilizador escolheu , assim como os sumários, títulos e classificação de cada resultado. Estes dados foram utilizados para avaliar a *performance* do sistema e para comparar com os resultados obtidos do mecanismo baseado nos perfis dos utilizadores.

De forma a avaliar o sistema, foram postos à prova 5 utilizadores, familiares com mecanismos de pesquisa. Foi pedido a estes para escreverem cinco textos diferentes com a ajuda do sistema de pesquisa.

Como conclusões do estudo, foi perceptível que a utilização do contexto proveniente das páginas *web* é muito mais enriquecedor para a construção do perfil do utilizador, do que a utilização de contexto proveniente de documentos *Word*[19], sendo importante diferenciar os pesos atribuídos a cada tipo de contexto.

Após essa diferenciação, foram obtidos resultados que eram cerca de 15% melhores que os resultados provenientes do *Google*. Este factor é importante, pois é um indicador de que é possível adaptar os mecanismos de pesquisa aos interesses do utilizador.

[16] http://windows.microsoft.com/pt-pt/windows/home
[17] http://office.microsoft.com/
[18] http://msn.com/
[19] http://office.microsoft.com/word/

2.5.2 Pesquisa Baseada em Contexto no Desenvolvimento de *Software*

O uso de contexto abrange diferentes áreas, sendo que neste trabalho Antunes [31] optou por incorporar contexto no ambiente de um programador. Para isso foram concentrados os esforços no código fonte que rodeia o utilizador quando este se encontra no *workspace* do seu IDE.

O grande objetivo deste projeto era, portanto, implementar um processo de pesquisa, que utilizava o contexto do desenvolvedor para devolver os vários artefatos de código ao mesmo. Esses artefatos que estão presentes no *workspace* do programador são representados a partir de ontologias de base estrutural e lexical. A primeira visa representar os artefatos e as várias relações que existem entre eles. A ontologia de base lexical representa os termos que são utilzados para referenciar esses artefatos e como é que esses termos estão relacionados entre eles. Relativamente à informação contextual do programador, esta é armazenada num modelo que representa o contexto do utilizador a um determinado momento, semelhante ao trabalho realizado por Challam [30] que foi referido na secção anterior, onde o contexto era armazenado até ser efetuada uma pesquisa.

Foi implementado um processo de pesquisa baseado em contexto, que faz uso do modelo de contexto para, posteriormente, influenciar, positivamente, a classificação dos resultados obtidos pela pesquisa. A relevância associada a cada resultado tem três tipos de classificações:

- ***relevance score*:** Representa a classificação atribuída ao resultado da pesquisa, sendo esta baseada no **modelo vetorial**
- ***structural score*:** Indica a relevância do resultado da pesquisa em relação ao contexto estrutural
- ***lexical score*:** Indica a relevância do resultado da pesquisa em relação ao contexto lexical

Estes três componentes são combinados com um conjunto de pesos, dando origem à classificação final para um resultado da pesquisa. O intervalo de classificação foi definido entre 0 e 1. O cálculo dos pesos teve como base um processo de aprendizagem que adapta os pesos de acordo com o comportamento do programador. Foi assumido que quando era selecionado um dos resultados da pesquisa, este era relevante para o utilizador, alterando desta forma o valor dos pesos.

Para avaliar o desempenho da aplicação, esta foi implementada no Eclipse IDE[20], de modo a provar que este tipo de pesquisa baseada em contexto poderia ajudar os utilizadores a encontrar informação relevante sobre o seu código, de uma forma mais eficiente. O processo de avaliação

[20]http://www.eclipse.org/

foi dividido em duas fases. A primeira consistia em cinco programadores, provenientes de uma empresa de *software* e a segunda consistia em quatro programadores de empresa, e seis com uma graduação em Engenharia Informática. O resultado desta experiência ofereceu, em média resultados positivos, sendo que deste trabalho é possível chegar à conclusão que o uso de contexto também pode ser incorporado em ambientes de programação, providenciando uma melhor experiência ao programador.

2.5.3 Recolha de Informação Baseada em Semântica e Sensível ao Contexto

Acreditando que o processo de IR só tem a ganhar com a introdução de contexto do utilizador, Carsten Kessler [32], apresentou dois aspectos que seriam fundamentais na implementação de um sistema de IR baseado em contexto. Kessler dizia que o *design* dos sistemas baseados em contexto necessitava de uma identificação do quê que era importante para o processo de recolha, ou seja, algo que permitisse saber que informação era parte do contexto do utilizador.

Foi avaliado que no processo de IR, a relevância dos resultados era determinada pelos aspectos contextuais presentes, sendo que ao realizar a mesma pesquisa em variados contextos, esta apresentaria uma ordem de resultados diferente. A partir daqui, foi constatado que ao invés de analisar a relevância dos resultados em relação à *query*, como é feito habitualmente, poderia pegar-se nos resultados diferentes e compará-los entre si, sendo assim uma forma alternativa de tentar perceber o impacto do contexto de cada um dos utilizadores nos resultados da pesquisa. Nesta tese foi então proposta uma medida denominada por *cognitively plausible Dissimilarity measure for Information Retrieval Results* (DIR) que apenas se baseia nos resultados da pesquisa, permitindo assim aos desenvolvedores identificar que aspetos contextuais são muito influentes no processo de recolha de informação, isto é, o quanto é que o utilizador humano influencia as diferenças no *ranking* dos resultados.

Outro ponto focado neste trabalho foi o facto de os aspetos contextuais terem de serem modelados de uma forma a suportarem interação com ontologias, bases de dados, entre outros. Sendo assim, Kessler na sua tese apresenta uma proposta de regras semânticas que permitem fazer integração com uma fonte muito rica em contexto, a Web Sensorial. De forma a modelar as preferências dos utilizadores baseados foi utilizada uma linguagem conhecida, *Semantic Web Rule Language* (SWRL) [33], sendo que esta suporta integrar relações com os diferentes sensores reais. No caso deste trabalho foram avaliadas condições meteorológicas em locais de *surf* face ao perfil de um utilizador e assim foram apresentados os locais por uma ordem que mais se adaptava ao perfil deste.

Nesta primeira fase foi possível, a partir da DIR, assegurar que a informação que era relevante para o perfil do utilizador era tida em conta, assim como era reduzida a complexidade das interfaces. Após os testes feitos pelos dois participantes foi possível constatar que estes revelaram uma forte correlação entre as análise dos participantes e os cálculos da DIR, tendo sido assim considerada como uma medida cognitiva e plausível para análise humana de resultados de *Information Retrieval*.

Relativamente à implementação das regras SWRL, estas permitiram fazer a integração dos sensores reais com o perfil ontológico do utilizador e assim apresentar resultados ao mesmo que seriam mais adequados.

2.5.4 Perfis Ontológicos do Utilizador para Pesquisa Personalizada

Neste trabalho, Ahu Sieg [34], apresenta uma proposta inovadora para a criação de perfis ontológicos de utilizadores, atribuindo pontuações a conceitos existentes numa ontologia, isto é, utilizarem o contexto proveniente do utilizador para re-organizarem a ordem de resultados face a uma determinada pesquisa. O contexto de cada utilizador é representado como uma instância de uma ontologia, sendo que os conceitos presentes nele têm pesos atribuídos (pontuações) que são atualizados de acordo com o comportamento do utilizador face aos mesmos, isto é, cada vez que o utilizador seleciona ou visualiza novos documentos. Tendo em conta o perfil ontológico do utilizador, os interesses que posteriormente acabam por surgir teriam de ser encontrados com o mínimo de intervenção humana possível. Para isso, Sieg, estipulou vários fatores que serviram de heurísticas para a captura de novos interesses tais como, frequência de visitas a uma página, o tempo que passou numa página, assim como a classificação de páginas como favoritas.

Como tal, a partir deste perfil, era pretendido personalizar os resultados da pesquisa re-ordenando os resultados de acordo com os interesses do utilizador. Foram necessários dois fatores principais: uma lista de resultados e os pesos atribuídos a cada interesse do utilizador. A computação da lista de resultados passou então por duas etapas, a primeira para encontrar a primeira lista de resultados, em que a sua ordem deve-se ao processo de similaridade de co-senos entre o documento e a *query*, sendo de seguida computada a similaridade entre o documento e cada conceito existente no perfil ontológico do utilizador. Quando fosse encontrado o conceito com maior similaridade, era atribuído ao documento uma pontuação calculada a partir da multiplicação entre o peso do conceito, a similaridade do documento com a *query* e a similaridade do conceito com a *query*.

Para demonstrarem os melhoramentos que podiam ser feitos com a re-ordenação de resultados, foram utilizadas *queries* pouco extensas, pois são as que tendem a ser mais ambíguas. Foram

conseguidos resultados positivo, a partir da introdução de perfis ontológicos dos utilizadores, sendo que estes permitiram desambiguação de termos.

2.5.5 Refinamento de Queries Baseado em Atividades para Recolha de Informação Sensível ao Contexto

Com o constante crescimento da utilização de dispositivos móveis, da computação móvel e da quantidade de informação que está presente na *web*, é possível ter acesso a informação em qualquer lugar e a qualquer hora. Para este trabalho, Hattori [35], no sentido de combater o tipo de pesquisas que são feitas em ambientes móveis (*queries* curtas e ambíguas), decidiu proceder ao desenvolvimento de um método que fosse capaz de refinar a *query* através de contexto real, como por exemplo, localização geográfica e atividades normalmente feitas nessas localizações através de técnicas de *Data Mining* [36].

O processo apresentado para esse aprimoramento, passa pelos seguintes passos:

- Numa determinada localização, o utilizador móvel faz a sua consulta
- É obtida a localização geográfica do utilizador (latitude e longitude) a partir do GPS ou então os locais onde este se encontra partir de RFID *tags*
- Passa-se à conversão do contexto posicional do utilizador para palavras contextuais (locais), a partir de um sistema de informação geográfica
- As ações e objetos dessas localizações são obtidas a partir de técnicas de *Data Mining*
- A cada palavra contextual (locais, ações, objetos) é atribuído um peso que determina a sua utilidade para a expansão da *query*
- São geradas *queries* alternativas a partir da *query* original e da palavra contextual com maior peso, permitindo assim ao utilizador escolher a que melhor se adapta a si
- Outra alternativa é apresentar os resultados da *query* com peso maior

A partir desta experimentação, foi possível constatar que a introdução deste método permitiu aumentar a precisão em vinte páginas pesquisadas pelo utilizador.

Capítulo 3

Análise e Especificação

Neste capítulo, primeiramente é apresentado uma visão geral sobre o sistema desenvolvido de nome *Mobile Semantic Context-Based Search* (MSCS), sendo de seguida apresentados e detalhados os casos de uso e requisitos do mesmo. São também descritas as tecnologias utilizadas, assim como a arquitetura definida na implementação do sistema. Com estas secções é possível obter uma clarificação do modo de funcionamento do sistema, assim como da relação entre os vários elementos intervenientes do sistema e da sua implementação.

3.1 Âmbito do Sistema Desenvolvido

O MSCS assenta num sistema de pesquisa semântica para ambientes móveis e que faz uso do contexto do utilizador para lhe providenciar melhores resultados. Este é constituído por uma aplicação móvel e um servidor.

Inicialmente o utilizador instala a aplicação móvel no seu dispositivo, de forma a que possa ser extraída deste, informação necessária para a construção de uma ontologia representativa do seu contexto, como por exemplo, mensagens de texto e o *email*. Após essa recolha de informação, a construção do contexto do utilizador é processada pelo servidor como será detalhado mais à frente.

Depois de construído o seu contexto, o utilizador pode proceder a pesquisas sobre um *dataset* de artigos da DBpedia, em que os interesses armazenadas na sua ontologia, vão influenciar a apresentação dos resultados na aplicação móvel, isto é, os primeiros elementos da lista irem ao encontro dos interesses do utilizador. A figura 3.1 visa representar o funcionamento geral deste sistema.

Figura 3.1: Visão geral do funcionamento do sistema MSCS.

3.2 Definição dos Casos de Uso

Dentro desta secção são definidos os casos de uso do sistema, sendo neste caso apenas definidos para o utilizador da aplicação móvel do sistema. Assim, pretende-se demonstrar quais são as funcionalidades da aplicação móvel disponíveis para o utilizador.

Na figura 3.2, está representado o diagrama de casos de uso para o utilizador da aplicação móvel.

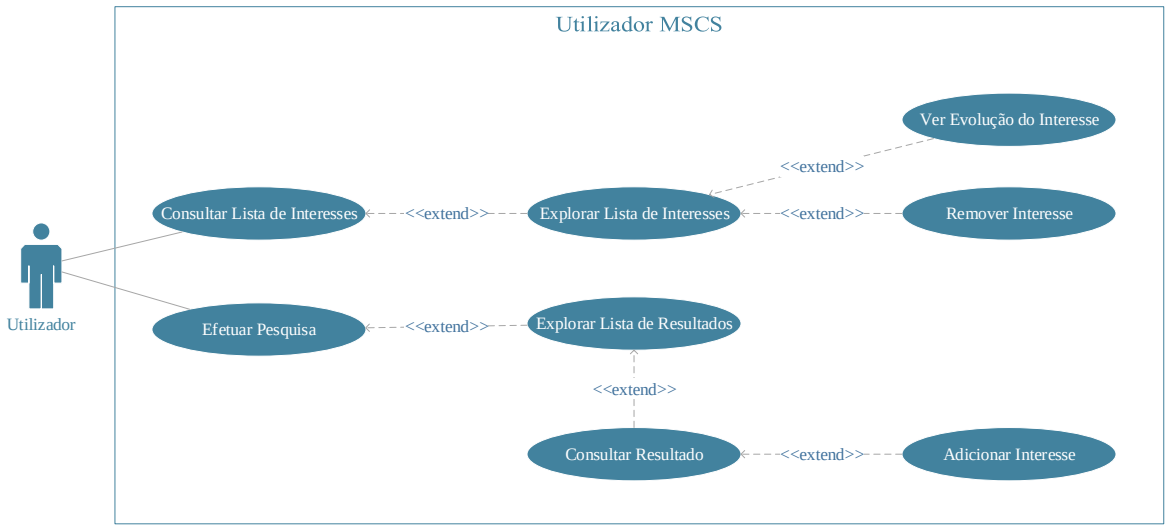


Figura 3.2: Representação dos casos de uso do utilizador da aplicação móvel.

Para uma melhor percepção dos casos de uso do utilizador é apresentado na tabela 3.1 uma descrição de cada um deles.

ID	Caso de uso	Descrição
CDU-1	Efetuar pesquisa	O utilizador deve poder efetuar as suas pesquisas através da aplicação móvel.
CDU-2	Explorar lista de resultados	O utilizador deve poder visualizar os resultados das pesquisas na aplicação móvel.
CDU-3	Consultar resultado	O utilizador seleciona um dos resultados para visualizar mais informação sobre este.
CDU-4	Adicionar interesse	O utilizador ao selecionar um resultado está a adicioná-lo aos seus interesses para enriquecimento do contexto.
CDU-5	Consultar Lista de Interesses	O utilizador deve poder solicitar a sua lista de interesses através da aplicação móvel.
CDU-6	Explorar lista de interesses	O utilizador deve poder visualizar a sua lista de interesses na aplicação móvel.
CDU-7	Visualizar evolução do interesse	O utilizador ao selecionar um interesse deve poder visualizar a evolução deste no contexto.
CDU-8	Remover interesse	O utilizador ao selecionar um resultado deve poder removê-lo do seu contexto.

Tabela 3.1: Descrição dos casos de uso do utilizador da aplicação móvel.

3.3 Definição dos Requisitos do Sistema

No que diz respeito aos requisitos do sistema, estes foram descritos através de tabelas para uma simples abordagem. Para além disso, os requisitos foram divididos de acordo com as duas entidades principais do sistema MSCS.

3.3.1 Aplicação Móvel

Os requisitos da aplicação móvel consistem nas operações que a aplicação móvel tem de realizar quando está a interagir com o utilizador ou a comunicar com o servidor. Para um melhor entendimento dos requisitos internos da aplicação móvel, é descrito na tabela 3.2 em quê que consiste cada um deles.

ID	Requisito	Descrição
REQ-1	Detetar SMS enviada Detetar SMS recebida	A aplicação que se encontra em *background* deve detetar quando uma SMS é recebida ou enviada.
REQ-2	Processar SMS	A aplicação deve fazer as operações necessárias à SMS para extrair informação e enviá-la para o servidor.
REQ-2.1	Recolher texto da SMS	Quando é detetada uma SMS, a aplicação deve extrair o texto da SMS para este ser enviado para o servidor.
REQ-2.2	Recolher *timestamp* da SMS	Quando é detetada uma SMS, a aplicação deve recolher o *timestamp* da SMS para este ser enviado para o servidor.
REQ-3	Enviar contexto SMS para o servidor	A aplicação após fazer o processamento da SMS deve enviar a informação de contexto para o servidor para que esta seja processada.
REQ-3.1	Enviar IMEI	A aplicação deve poder enviar o IMEI do *smartphone* na informação de contexto de forma a identificar o utilizador no servidor.
REQ-3.2	Enviar credenciais Gmail	A aplicação deve poder enviar as credenciais Gmail do utilizador para assim recolher contexto dos Emails no servidor.
REQ-3.3	Enviar texto da SMS	A aplicação deve poder enviar o texto da SMS na informação de contexto para serem extraídos tópicos de interesse no servidor.
REQ-4	Enviar contexto para o servidor	Ao ser feito algum pedido ao servidor, a aplicação deve enviar a informação de contexto para o servidor para que esta seja processada. **Inclui os requisitos 3.1, 3.2 e 3.3.**
REQ-5	Enviar pedido de pesquisa para o servidor	A aplicação, após o pedido de pesquisa feito pelo utilizador, deve enviar a *query* para o servidor para que esta seja interpretada.
REQ-6	Mostrar resultados ao utilizador	A aplicação, após receber a resposta do servidor, deve apresentar a lista de resultados que satisfaz o pedido do utilizador.

REQ-6.1	Mostrar mais informação do resultado	A aplicação, caso o utilizador tenha selecionado algum resultado da lista, deve mostrar ao utilizador mais informação sobre o mesmo.
REQ-6.2	Adicionar/Atualizar o interesse do utilizador	A aplicação, caso o utilizador tenha visualizado mais informação sobre algum resultado, deve ser enviado para o servidor o resultado para ser adicionado aos interesses ou então reajustado.
REQ-7	Enviar pedido de interesses para o servidor	A aplicação, após o pedido de interesses feito pelo utilizador, deve enviar para o servidor a solicitação para que este os recolha.
REQ-8	Mostrar interesses ao utilizador	A aplicação, após receber a resposta do servidor, deve apresentar a lista de interesses do utilizador.
REQ-8.1	Mostrar evolução do interesse	A aplicação, caso o utilizador tenha selecionado algum interesse da lista, deve mostrar ao utilizador a evolução deste.
REQ-8.2	Remover o interesse do utilizador	A aplicação, caso o utilizador tenha selecionado para remover um interesse, deve enviar para o servidor o pedido de remoção do mesmo.

Tabela 3.2: Descrição dos requisitos internos da aplicação móvel.

3.3.2 Servidor

Relativamente ao servidor, foram definidas as funcionalidades internas do mesmo, de forma a entender que tipo de operações este deve realizar.

A descrição de cada um dos requisitos internos do servidor é apresentada na tabela 3.3, para que desta forma, seja entendido os processos feitos pelo mesmo.

ID	**Requisito**	**Descrição**
REQ-9	Receber o contexto SMS	O servidor deve receber o contexto das SMS para futura extração de interesses.
REQ-9.1	Extrair interesses	O servidor deve extrair interesses das SMS e *emails* para serem adicionados na ontologia.

REQ-9.2	Inserir novo interesse na ontologia	O servidor deve inserir o interesse, caso este ainda não exista, na ontologia de perfil do utilizador.
REQ-9.3	Atualizar o peso do interesse na ontologia	O servidor, ao extrair um interesse, deve atualizar o peso do mesmo na ontologia de acordo com o seu número de ocorrências.
REQ-10	Recolher o contexto do *email*	O servidor deve monitorizar a conta Gmail do utilizador no sentido de extrair contexto dos *emails* do utilizador.
REQ-10.1	Recolher assunto do *email*	O servidor, ao monitorizar a conta Gmail do utilizador, deve recolher o assunto dos *emails* para serem extraídos interesses. **Inclui os requisitos 9.1, 9.2 e 9.3.**
REQ-10.2	Recolher corpo do *email*	O servidor, ao monitorizar a conta Gmail do utilizador, deve recolher o corpo dos *emails* para serem extraídos interesses. **Inclui os requisitos 9.1, 9.2 e 9.3.**
REQ-11	Receber o contexto do utilizador	O servidor deve receber o contexto do utilizador para validar a sua autenticação.
REQ-11.1	Registar o utilizador na ontologia	O servidor quando recebe contexto do utilizador, deve registá-lo na ontologia caso ainda não esteja.
REQ-12	Processar pedidos de pesquisa	O servidor deve conseguir processar os pedidos de pesquisa do utilizador, feitos a partir da aplicação móvel.
REQ-12.1	Procurar no índice da DBpedia	O servidor deve realizar as pesquisas sobre o índice da DBpedia em conjunto com a ontologia de perfil do utilizador.
REQ-13	Enviar resultados para a aplicação móvel	O servidor deve enviar os resultados da pesquisa do utilizador para a aplicação móvel para que estes sejam mostrados ao utilizador.
REQ-14	Processar classificações de interesses	Após a devolução de resultados ao utilizador, o servidor poderá ter de adicionar ou atualizar interesses da ontologia do utilizador. **Inclui os requisitos 9.2 e 9.3.**

REQ-15	Enviar interesses para a aplicação móvel	O servidor deve enviar os interesses do utilizador para a aplicação móvel para que estes sejam mostrados ao utilizador.
REQ-16	Remover o interesse da ontologia	Após a devolução de resultados ao utilizador, o servidor poderá ter de remover interesses da ontologia do utilizador.
REQ-17	Analisar interesses do utilizador	O servidor deve analisar os interesses do utilizador diariamente, no sentido de decrementar o seu peso ou remover o interesse.
REQ-17.1	Decrementar o peso do interesse	O servidor deve decrementar o peso do interesse, caso este se encontre sem modificações há mais de 24 horas.
REQ-17.2	Limpar o interesse da ontologia	O servidor deve eliminar o interesse do utilizador da ontologia, caso ao decrementar o peso deste atinga um valor menor que o mínimo estipulado.

Tabela 3.3: Descrição dos requisitos internos do servidor.

3.4 Arquitetura do Sistema

No que diz respeito à arquitetura do sistema, esta foi dividida em duas vertentes, sendo estas física e lógica. Desta forma é possível entender quais são os componentes necessários para o funcionamento do sistema e de que forma é que estes operam entre si.

Por fim é feita uma descrição das tecnologias escolhidas no desenvolvimento do sistema, e que se encontram representadas no *design* da arquitetura.

3.4.1 Arquitetura Física

Dentro desta subsecção pretende-se apresentar os vários componentes que constituem o sistema MSCS e, desta forma, descrever como é que estes comunicam entre si. Na figura 3.3, é possível então observar a constituição do sistema MSCS através de uma representação com duas perspetivas, cliente e servidor.

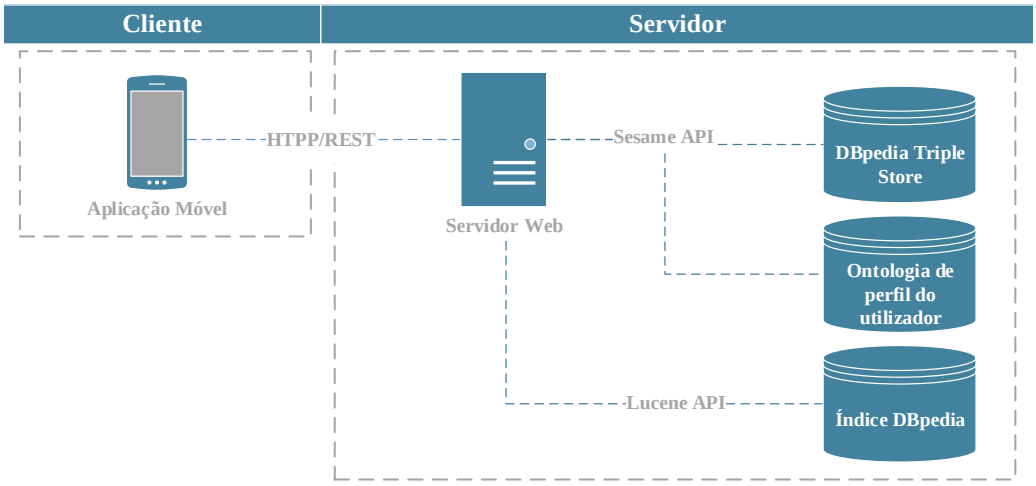


Figura 3.3: Representação da arquitetura física do sistema.

O lado do cliente é então composto por um dispositivo móvel *Android*, que têm por sua vez instalado uma aplicação móvel responsável pela interação do utilizador com o sistema. Assim, a partir desta é possível recolher o contexto do utilizador, recolher as credenciais Gmail, efetuar pesquisas sobre o índice da DBpedia e consultar os interesses do utilizador.

A comunicação entre a aplicação e servidor é efetuada a partir de um serviço REST, sendo que as estruturas de dados que são enviadas nos dois sentidos, encontram-se no formato *JSON*[1] que é bastante eficiente para comunicações móveis.

Por sua vez, o servidor encontra-se com duas API's principais implementadas. A primeira, *Sesame* API, responsável pela gestão dos triplos da DBpedia armazenados e da ontologia de perfil do utilizador. A segunda, *Lucene* API, é responsável pela indexação dos triplos da DBpedia para um acesso de maior *performance*.

3.4.2 Arquitetura Lógica

No que diz respeito à arquitetura lógica, esta pretende especificar como é que cada um dos componentes do sistema se encontra implementado, ao nível dos seus módulos de *software*. Este tipo de arquitetura, devido ao seu nível de detalhe, encontra-se dividido em dois ambientes, um para a aplicação móvel, outro para o servidor.

[1] http://www.json.org/

Aplicação Móvel

Na aplicação móvel optou-se por dividir a arquitetura da mesma através de diferentes camadas, no sentido de simplificar a sua descrição. A figura 3.4, visa demonstrar a modelação da aplicação móvel através de três camadas (persistência, lógica de negócio, apresentação).

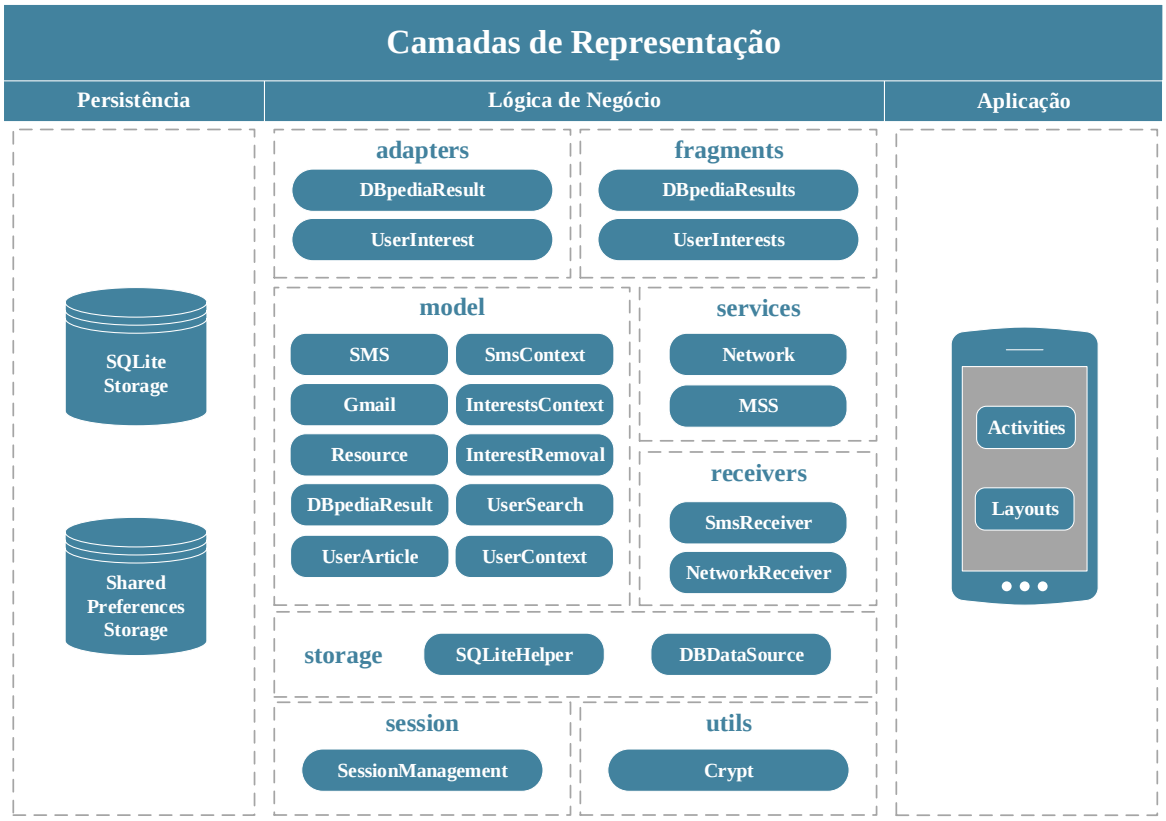


Figura 3.4: Representação da arquitetura lógica da aplicação móvel do sistema.

Na camada de persistência estão presentes os tipos de armazenamento utilizados para guardar os dados do utilizador no dispositivo. Estes dados podem ser SMS's do utilizador ou então os dados de autenticação do mesmo. Para um melhor entendimento são descritos os dois tipos de armazenamento:

- ***SQLite Storage:*** Este tipo de armazenamento é responsável por guardar as mensagens de texto do utilizador, quando estas não são bem sucedidas no envio para o servidor, maior parte das vezes, por falta de ligação à *Internet*. Para guardar estes dados localmente, utiliza-se uma base de dados SQL, em que para a criação desta, recorreu-se à biblioteca *SQLite*[2].

[2]http://www.sqlite.org/

- ***Shared Preferences Storage:*** Aqui já não é necessário recorrer a uma estrutura, sendo que este serve apenas para guardar alguns dados de sessão do utilizador. Estes são constituídos pelo IMEI do dispositivo móvel, as credenciais Gmail do utilizador e um indicador de que este já tem o seu perfil criado no servidor. Assim, estes dados ficam armazenados nas *Shared Preferences*[3] do dispositivo.

Para a lógica de negócio são apresentados os vários módulos do sistema desenvolvido, ou seja, os *packages* da aplicação móvel que permitem à mesma cumprir os requisitos definidos. A descrição seguinte especifica o que cada um destes é responsável por fazer:

- ***adapters:*** Aqui encontram-se implementados os *Android Adapters*[4] que permitem estruturar a apresentação dos elementos constituintes das listas de resultados e interesses.
- ***fragments:*** Aqui são implementados os *Android Fragments*[5] responsáveis por apresentar as duas listas referidas anteriormente, ou seja, *DBpediaResultsFragment* para os resultados das pesquisas e *UserInterestsFragment* para os interesses do utilizador.
- ***model:*** Responsável por definir as estruturas de dados que são instanciadas para o bom funcionamento da aplicação móvel, podendo estas ser mensagens de texto, credenciais Gmail, resultados de pesquisas, entre outros.
- ***receivers:*** Responsável por funcionamento *background*, este módulo tem implementado dois *BroadcastReceivers*[6]. O primeiro (*SmsReceiver*), encontra-se à escuta de mensagens de texto recebidas pelo utilizador. Quanto ao *NetworkReceiver* este verifica constantemente se existe ligação à *web*, para que sejam efetuadas operações em espera, como por exemplo, envio de SMS's armazenadas.
- ***services:*** Também responsável pelo funcionamento *background* da aplicação móvel. O *MSSService* está encarregue de inicializar a aplicação, sendo que é nesta classe também inicializado um *listener* que fica à escuta de mensagens enviadas, pois o *Android* não dispõe de um *BroadcastReceiver* para recolher mensagens enviadas. Foi utilizada a classe *SMSObserver* que faz *extends* da classe *ContentObserver* para implementar este *listener*. O *NetworkService* é responsável pelo envio de pedidos para o servidor, como por exemplo, pesquisas, SMS's para extração, entre outros. Para a realização destes pedidos, são utilizadas *AsyncTasks*[7] para que a aplicação continue a responder a outros pedidos.

[3]http://developer.android.com/guide/topics/data/data-storage.html#pref
[4]http://developer.android.com/guide/topics/ui/declaring-layout.html#AdapterViews
[5]http://developer.android.com/guide/components/fragments.html*text*
[6]http://developer.android.com/reference/android/content/BroadcastReceiver.html
[7]http://developer.android.com/reference/android/os/AsyncTask.html

- ***storage:*** Torna-se responsável por aceder e gerir o armazenamento definido na camada de persistência como *SQLite Storage*. Desta forma, é possível manipular a base de dados presente no dispositivo móvel, como inserir, pesquisar e remover dados da mesma. Estas operações são realizadas a partir da classe *DBDataSource*.

- ***session:*** Este módulo é encarregue de gerir os dados de sessão do utilizador, daí atuar sobre o *Shared Preferences Storage* da camada de persistência. Com a implementação deste módulo os dados de sessão do utilizador armazenados no *smartphone* estão acessíveis em qualquer parte da aplicação a partir da classe *SessionManagement*.

- ***utils:*** Utilizado para auxiliar no funcionamento de outros módulos, sendo que a função mais importante é o facto de ser utilizado para encriptar/desencriptar a *password* Gmail do utilizador. Também pode ser usado para efetuar o mesmo processo com o texto das SMS's.

Por fim, a camada de apresentação é constituída pelas *Activities* e *layout's* definidos que permitem que o utilizador possa interagir com o sistema, desde o efetuar pesquisas, consultar interesses, consultar artigos da DBpedia, entre outros. Estas interações serão demonstradas no próximo capítulo.

Servidor

Para uma representação lógica da arquitetura do servidor, seguiu-se, mais uma vez, uma divisão por módulos, sendo que a aplicação servidor tem integrada também outras aplicações para suportar funcionalidades auxiliares ao sistema, como por exemplo, operações de Processamento de Linguagem Natural (PLN) para a obtenção de interesses (KISWSNLP) e para a gestão da ontologia do utilizador (KISWSREPOSITORY).

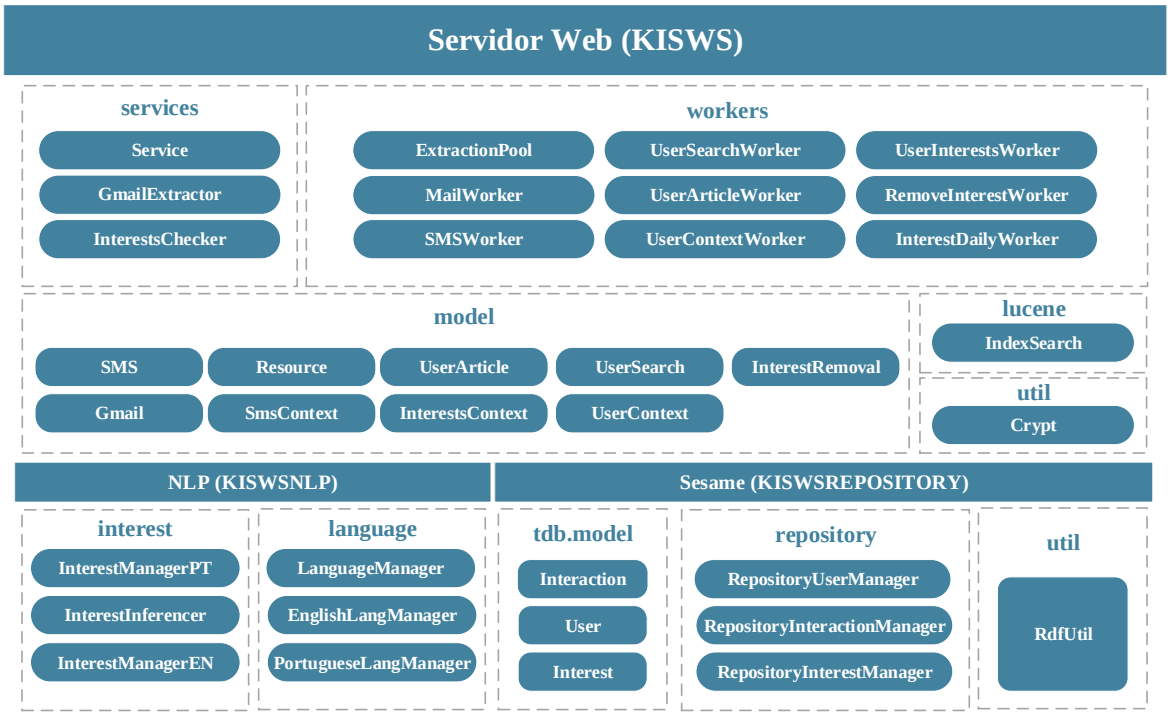


Figura 3.5: Representação da arquitetura lógica do servidor do sistema.

Numa camada *web* (KISWS), o servidor é caracterizado por módulos que permitem a comunicação entre o mesmo e a aplicação móvel, assim como módulos que permitem a estruturação dos dados que são transferidos entre os dois sistemas. Estes são divididos da seguinte forma:

- ***services:*** Aqui está implementado o serviço REST da aplicação (*Service*), responsável por receber todos os pedidos que são efetuados pela aplicação móvel ao servidor como por exemplo, pedidos de pesquisa, pedidos de lista de interesses, entre outros. Para além disso, encontram-se implementados os serviços *Quartz*[8], *GmailExtractor* e *InterestsChecker*, que são responsáveis por recolher os *emails* do utilizador e realizar a verificação de inatividade dos interesses, respetivamente.

- ***workers:*** Para este módulo estão implementados os *workers* para cada um dos tipos de pedidos que o servidor tem de suportar, desde o processamento de pesquisas, extração de interesses de *emails*/SMS's, remoção de interesses, entre outros. Para a execução destes *workers*, está definido um *Executor Service* (ES) por cada tipo de pedido, que tem a seu dispôr uma *Thread Pool* e uma *queue*, sendo feita a gestão das *workers threads* a partir destes elementos. O funcionamento deste módulo vai ser detalhado no próximo capítulo.

- ***model:*** Este é responsável por definir as estruturas de dados que estão presentes no

[8]http://quartz-scheduler.org/

servidor. À semelhança da aplicação móvel, estas estruturas de dados podem ser mensagens de texto, credenciais Gmail, resultados da pesquisa no índice da DBpedia, entre outros.

- ***lucene:*** Neste módulo está implementada a classe responsável por realizar as pesquisas sobre o índice da DBpedia. Estas são realizadas quando é necessário proceder a uma associação de interesse ou então quando é feita uma pesquisa a partir da aplicação móvel.
- ***util:*** Assim como na aplicação móvel, este módulo é utilizado para encriptar/desencriptar os dados da conta Gmail do utilizador.

O servidor, no sentido de conseguir extrair interesses do contexto do utilizador, para de seguida estes serem associados a artigos da DBpedia, realiza um conjunto de operações, onde as técnicas de PLN são fundamentais. Para isso, este tem integrado uma aplicação responsável por essas operações (KISWSNLP), encontrando-se esta dividida em dois módulos:

- ***interest:*** Neste módulo estão implementadas as funcionalidades que permitem extrair os tópicos de interesse dos emails e sms's dos utilizadores. Este processo está implementado para a língua portuguesa e inglesa (*InterestManagerPT/InterestManagerEN*). Para além disso, a classe *InterestInferencer* é responsável por inferir os interesses construídos, no modelo de tópicos da DBpedia que foi criado. No capítulo seguinte é possível verificar o seu funcionamento.
- ***language:*** Aqui estão definidas as classes que permitem identificar se o texto extraído está na língua inglesa ou portuguesa.

Relativamente ao repositório responsável por guardar o contexto do utilizador, este é gerido a partir da aplicação KISWSREPOSITORY, em que esta utiliza a API do *Sesame* para realizar operações sobre a mesma. Assim, a aplicação encontra-se dividida em três módulos principais:

- ***tdb.model:*** Neste módulo estão definidas as estruturas de dados que são armazenadas no repositório, sendo estas *User, Interaction, Interest.*
- ***repository:*** Responsável por fazer a gestão das estruturas intervenientes do repositório do utilizador, como por exemplo, obter interesses do utilizador, adicionar novos interesses do utilizador, atualizar interesses, entre outros.
- ***util:*** O módulo *util* apresenta algumas funcionalidades auxiliares para a gestão do repositório, como por exemplo, transformar as entidades do repositório em instâncias das classes descritas no módulo *tdb.model.*

Ontologia de Perfil do Utilizador

O projeto KISWSREPOSITORY referido anteriormente é responsável pela gestão da ontologia do utilizador, sendo que esta está estruturada de forma a conseguir armazenar os interesses e outros dados dos utilizadores, como por exemplo, IMEI, credenciais, entre outros. Na figura 3.6 é demonstrado como é que a ontologia se encontra definida.

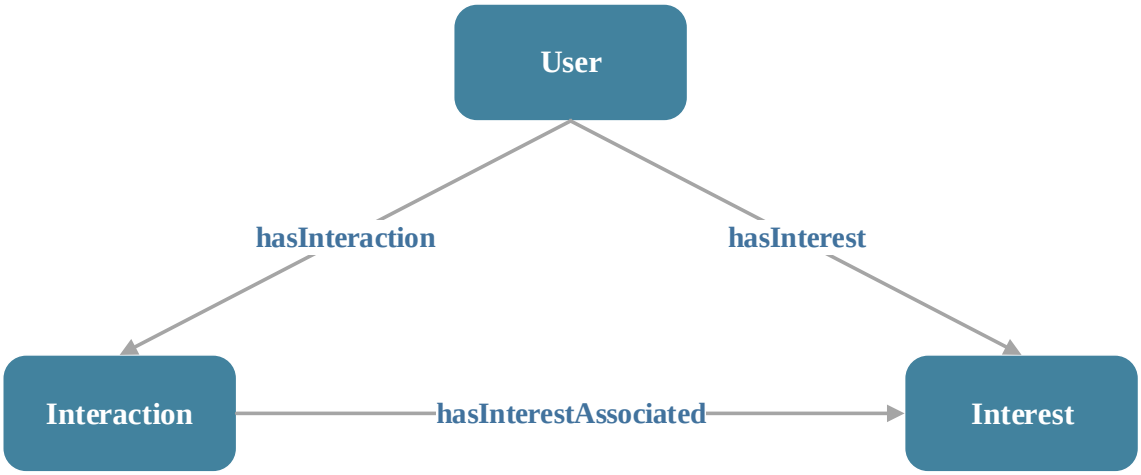


Figura 3.6: Ontologia de perfil do utilizador.

As suas entidades principais podem ser descritas da seguinte forma:

- ***User:*** Esta classe é responsável por identificar um utilizador no sistema. As propriedades que esta classe tem associadas são o ID do utilizador, o IMEI, o *email* e respetiva *password*. Para além disso, a classe *User* pode ter associado *Interactions* e *Interests*.
- ***Interaction:*** Uma *Interaction* é criada quando são extraídos interesses de *emails* ou SMS's, ou seja, uma *Interaction* identifica um *email* ou SMS. Aqui é guardado o ID do utilizador a que se encontra associada, assim como a data de criação. Esta classe deve ter um ou mais interesses associados.
- ***Interest:*** A classe *Interest* é responsável por identificar um interesse do utilizador. Aqui são armazenadas propriedades como, a palavra ou conjunto de palavras que deram origem ao interesse, o título do artigo que foi associado, o peso do interesse, a língua do interesse, assim como o ID do utilizador a que este se encontra associado. Uma instância da classe *Interest* é criada de duas formas: quando são criadas *Interactions*, pois estas vão ter sempre associadas pelo menos uma instância, ou quando é consultado na aplicação um novo artigo, sendo que este vai surgir como *Interest* associado à classe do utilizador, ao contrário de ser associado a uma *Interaction*.

3.4.3 Tecnologias e Ferramentas Utilizadas

Neste tópico vão ser descritas as tecnologias que foram necessárias para o desenvolvimento deste sistema, assim como as ferramentas que auxiliaram na integração dessas tecnologias.

Visto que o sistema desenvolvido consiste em comunicação cliente-servidor e se trata de classificação de informação real, isto é, o contexto do utilizador, foram necessárias precauções no que diz respeito à escolha de algumas destas tecnologias e ferramentas.

Como tal, estas tecnologias foram adotadas das escolhas tecnológicas que já teriam sido feitas quando o projeto foi inicializado.

Aplicação Móvel

Como o conceito do sistema é extrair contexto do utilizador a partir do dispositivo móvel, foi decidido o desenvolvimento de um aplicação nativa *Android*[9]. A partir de uma aplicação nativa, o acesso à informação armazenada e necessária para construir contexto, torna-se muito mais simples, como por exemplo, a recolha de mensagens de texto.

Esta aplicação é suportada por dispositivos com a versão 2.3.3 ou superior do *Android* e para o seu desenvolvimento foi utilizado o *Android* SDK[10] juntamente com o *Eclipse* IDE.

Servidor

Para o desenvolvimento do servidor do sistema, foi escolhida a linguagem *Java*, assim como a orientação para a *web*. Esta escolha deveu-se ao facto de existirem diversas API's[11] para esta linguagem que suportam comunicação com o cliente, gestão de Emails e que suportam tecnologias da Web Semântica (WS). Para suportar a orientação à *web* foi utilizado o Jboss[12] AS 5 que suporta comunicação REST 37.

Armazenamento e Gestão dos Dados

Para a gestão dos dados, visto que estamos a trabalhar com tecnologias de *web* semântica, utilizou-se a API para Java do *Sesame*[13], visto que esta é simples de utilizar e, é poderosa no que diz respeito ao armazenamento de dados RDF e na pesquisa através de *queries* SPARQL.

[9] http://www.android.com/
[10] http://developer.android.com/sdk/index.html
[11] http://pt.wikipedia.org/wiki/API
[12] http://www.jboss.org/
[13] http://www.openrdf.org/

DBpedia

Para conseguir interpretar os interesses do utilizador, foi utilizado o projeto DBpedia[14] para que assim estes permitissem ao utilizador verificar que tipo de informação era possível obter a partir do seu contexto. O projeto DBpedia permite ter acesso à informação da *Wikipedia*[15] devidamente estruturada, sendo que para a realização desta dissertação foi carregada toda a DBpedia em PT/EN.

No entanto, como estamos a usar triplos da DBpedia, isto implica uma grande quantidade de informação. Para resolver este problema utilizou-se o *Apache Lucene*[16], no sentido de criar um índice de triplos da DBpedia, passando a pesquisa a ser feita sobre este para aumentar a *performance* do sistema.

A tabela 3.4 especifica como é que cada um dos artigos indexados se encontra estruturado.

Atributo	Descrição
URI	Identificador do artigo da DBpedia
Title	Título do artigo da DBpedia
Abstract	Resumo do artigo da DBpedia
Lang	Língua do artigo da DBpedia
Topics	Lista dos tópicos mais relacionados com o artigo da DBpedia

Tabela 3.4: Estrutura de um artigo da DBpedia indexado.

Como tal No próximo capítulo, encontra-se descrito como é que estes elementos indexados atuam sobre o sistema.

[14] http://dbpedia.org/
[15] http://www.wikipedia.org/
[16] https://lucene.apache.org/core/

Capítulo 4

Sistema Desenvolvido

Para o desenvolvimento desta dissertação, foi necessário implementar um protótipo que permitisse satisfazer os requisitos definidos. Dentro deste capítulo é detalhado o funcionamento do sistema de modo a que possam ser entendidas as soluções de implementação do mesmo. Como já foi apresentado, o sistema divide-se numa aplicação móvel e numa aplicação servidor. Assim, este capítulo divide-se em duas secções.

4.1 Aplicação Móvel

Relativamente à aplicação móvel, esta encontra-se desenvolvida para dispositivos com sistema operativo *Android* cuja versão seja 2.3.3 ou superior. Dentro desta secção são descritas as funcionalidades implementadas na aplicação *Mobile Semantic Context-Based Search* (MSCS).

4.1.1 Autenticação

Na primeira vez em que o utilizador interage com a aplicação móvel, é pedido a este que insira as suas credenciais Gmail (ver figura 4.1). Esta informação é essencial para o bom funcionamento do sistema caso contrário não é possível extrair interesses do utilizador a partir do seu *email*.

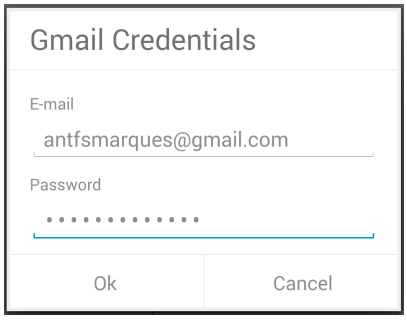


Figura 4.1: Inserção das credenciais Gmail através da aplicação móvel.

Após a primeira autenticação, estes dados, juntamente com o IMEI do dispositivo são processados pela classe *SessionManagement* do módulo *session* (camada de lógica de negócio) (ver subsecção 3.4.2) que procede ao armazenamento nas *Shared Preferences* do dispositivo (camada de persistência). De seguida, estes dados são enviados para o servidor, através do serviço *NetworkService* que contém uma *AsyncTask* responsável por tratar do pedido de registo do utilizador no sistema.

Para além disso, ao estarem guardados no dispositivo, os dados vão servir para identificar o utilizador quando é feito algum pedido ao servidor. A figura 4.2 demonstra como esta funcionalidade se encontra implementada.

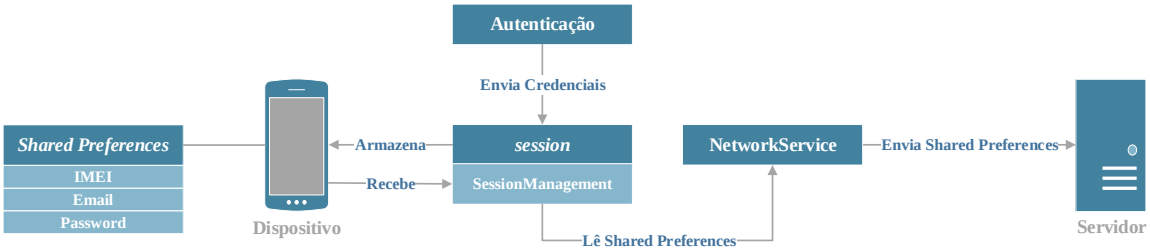


Figura 4.2: Diagrama representativo do armazenamento e posterior envio dos dados de autenticação.

4.1.2 Gestão de SMS's

Com a aplicação móvel instalada no dispositivo, o utilizador, para além das operações que estão ao seu dispôr, passa também a ter funcionalidades a correr em *background*, mais precisamente, *services* e *receivers* (ver subsecção 3.4.2), que se encontram à escuta de SMS's enviadas e recebidas. O objetivo destes é permitir que a aplicação consiga recolher essa informação e enviá-la para o servidor, enriquecendo assim o contexto do utilizador através da extração de interesses.

No entanto, a ligação à internet pode implicar perda de informação, isto é, mensagens perdidas. Desta forma, o *SQLite Storage* e o módulo *storage* especificados anteriormente (subsecção 3.4.2)

são fundamentais para lidar com este problema, pois através de uma estrutura de dados baseada em SQL é possível armazenar as mensagens no dispositivo.

A aplicação móvel, de forma a conseguir detetar as mensagens enviadas e recebidas, encontra-se com o SMS Receiver e o SMS Observer a executar em *background* (ver subsecção 3.4.2), com o objetivo de conseguir capturar essas mensagens.

Quando é detetada uma mensagem enviada ou recebida, é feita uma verificação de ligação à internet, caso não exista, a mensagem vai ser armazenada numa estrutura de dados baseada em SQL, através do módulo *storage*, caso contrário é enviada para o servidor pela *AsyncTask* respetiva, que se encontra implementada no *Network Service*.

No entanto, as mensagens armazenadas no dispositivo, são posteriormente enviadas para o servidor, quando é detetada ligação à internet (*Network Receiver*). Estas passam então pelo mesmo processo e são enviadas pela *AsyncTask*.

Através da figura 4.3 é possível entender como esta gestão é feita.

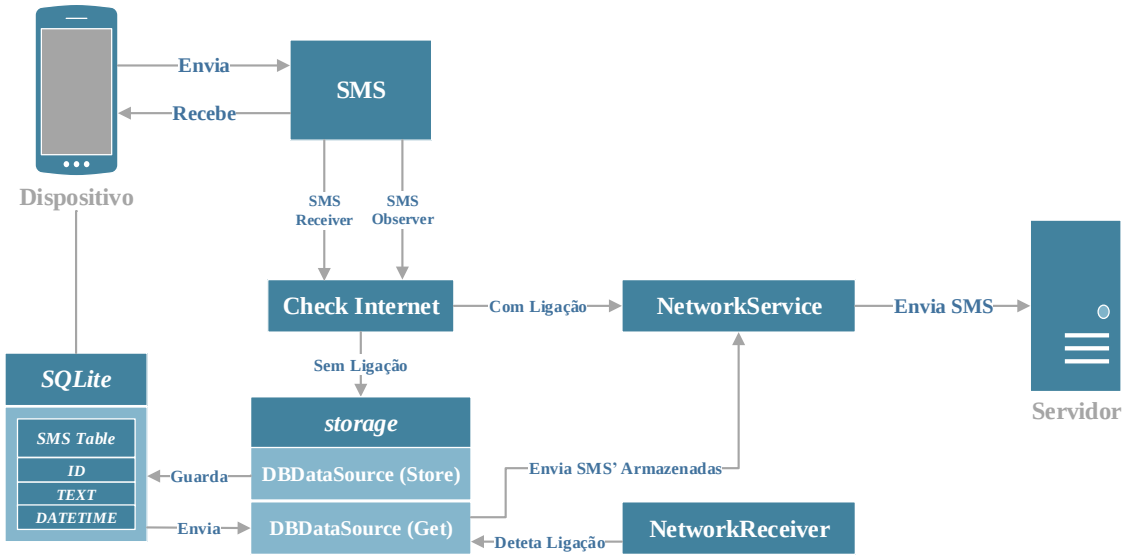


Figura 4.3: Diagrama representativo da gestão das SMS's no dispositivo.

4.1.3 Pesquisa de Artigos da DBpedia

A funcionalidade de pesquisa assume o papel principal, pois pretende-se que o utilizador consiga obter resultados de um repositório da DBpedia que vão ao encontro de aquilo que são os seus interesses. No âmbito da pesquisa, o utilizador tem três cenários possíveis tais como são demonstrados nas figuras 4.4, 4.5 e 4.6.

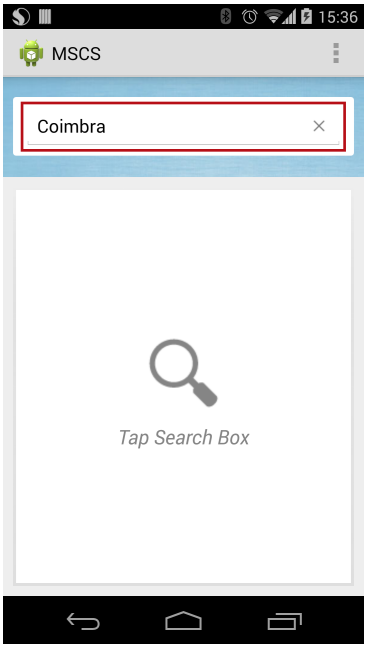


Figura 4.4: Ambiente da aplicação quando se pretende efetuar uma pesquisa.

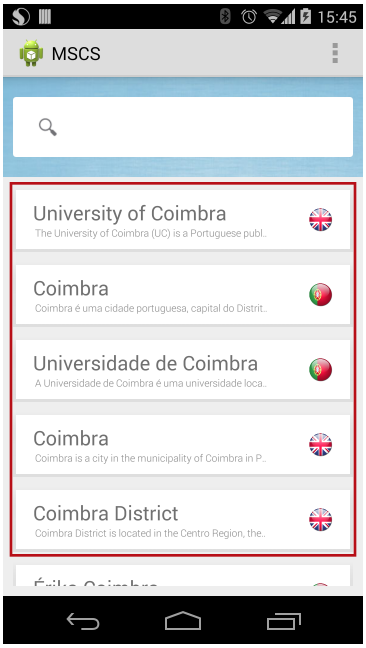


Figura 4.5: Apresentação dos resultados ao utilizador.

Figura 4.6: Visualização de um artigo devolvido pela pesquisa.

Dentro do primeiro cenário (figura 4.4), o utilizador deve introduzir a *query* na caixa de pesquisa. Após o processamento da pesquisa, é apresentada a lista de resultados no dispositivo (figura 4.5), sendo que estes estão ordenados pela seu peso no contexto da pessoa em questão, isto é, estão dispostos por ordem decrescente de peso.

Na visualização dos artigos resultantes da pesquisa, o utilizador pode ver para cada um deles, o título, um extrato do *abstract* e a língua, sendo que desta forma torna-se mais fácil saber qual o artigo que pretende consultar. Caso o pretenda fazer, basta carregar sobre o elemento da lista e acede à página da *Wikipedia* do artigo para visualizar mais informação sobre aquele tópico (ver figura 4.6). Para além disso, ao ser visualizado um artigo de interesse, o contexto é enriquecido, visto que é enviada informação para o servidor a indicar que o utilizador consultou aquele artigo. O pedido de pesquisa e a notificação de consulta do artigo, são enviados para o servidor a partir das *AsyncTasks* respetivas do serviço *NetworkService*.

4.1.4 Interesses do Utilizador

No entanto, o utilizador pode no seu dispositivo consultar os seus interesses atuais, sendo que a partir desta funcionalidade pode verificar a evolução de cada um deles, assim como eliminá-los caso não os considere como parte do seu contexto. Desta forma, está também a contribuir para os seus interesses. Nas figuras 4.7, 4.8 e 4.9 é demonstrada a interação do utilizador com os seus interesses.

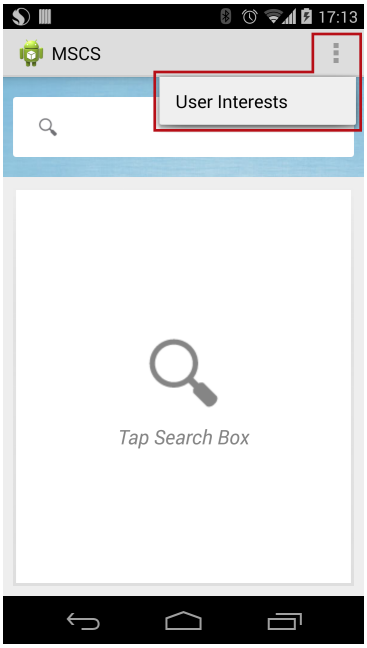


Figura 4.7: Botão que permite ao utilizador verificar os seus interesses.

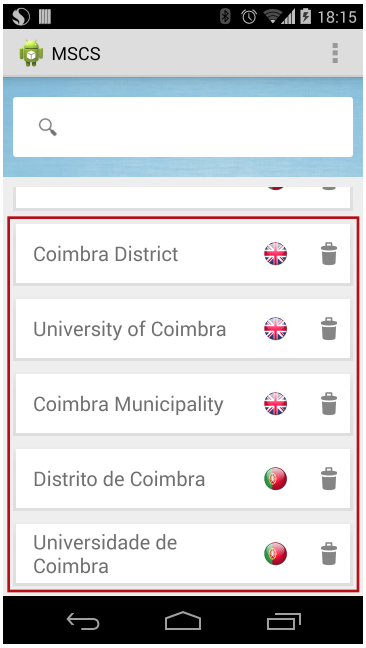


Figura 4.8: Apresentação dos interesses do utilizador.

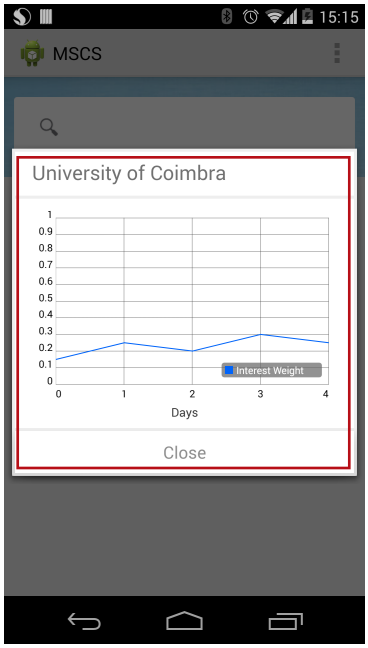


Figura 4.9: Análise da evolução do interesse ao longo dos dias.

Quando se encontra na aplicação móvel, o utilizador pode verificar os interesses que fazem parte do seu contexto atual. Para isso, basta carregar no botão *Action Overflow*[1] presente no canto superior direito da aplicação, sendo que este vai apresentar o *Action Button* denominado por *User Interests* (ver figura 4.7).

Ao pressionar este último botão, é apresentada a lista de interesses do utilizador (figura 4.8) onde este pode fazer duas operações. A primeira é eliminar o interesse, caso o utilizador não considere como adequado. Esta operação é feita a partir do ícone indicativo de lixo.

Relativamente à segunda funcionalidade, esta consiste no utilizador poder consultar a forma como interesse tem evoluído diariamente. Para tal, apenas é necessário carregar sobre um dos interesses e o gráfico é apresentado numa caixa de diálogo (ver figura 4.9). Para complementar esta funcionalidade foi utilizada a biblioteca *GraphView*[2] para a elaboração dos gráficos. É de referir, mais uma vez, que o pedido de lista de interesses e o pedido de remoção de interesse, são enviados para o servidor a partir das *AsyncTasks* implementadas no *NetworkService*.

4.1.5 Detalhes Técnicos

De forma a obter uma aplicação móvel de maior qualidade, houve cuidado na implementação da mesma. De seguida, são apresentadas algumas opções técnicas que serviram para um melhor

[1] http://developer.android.com/design/patterns/actionbar.html
[2] http://android-graphview.org/

funcionamento da aplicação.

Utilização de *Android Fragments*

Como a aplicação móvel desenvolvida tem um número reduzido de ambientes para o utilizador, isto é, de pesquisa e gestão de interesses, optou-se por não criar uma *Activity*[3] para cada um, e sim utilizar *Fragments*.

Esta gestão foi feita no sentido de tornar a aplicação mais leve e flexível para o utilizador, no que diz respeito a transições entre ambientes.

Inicialmente é apresentado ao utilizador o ambiente de pesquisa, em que está ativo o *DBpediaResultsFragment*, responsável por apresentar a lista de resultados da pesquisa. Quando o utilizador pretende aceder aos seus interesses, o *fragment* encarregue da lista de pesquisas é adicionado a uma *stack* responsável pela gestão dos *fragments* e passa a ser apresentado o *UserInterestsFragment*.

Desta forma, quando o utilizador se encontra a verificar os seus interesses, pode voltar facilmente à lista de pesquisas se pressionar o botão *back* do dispositivo. Caso o utilizador faça uma pesquisa quando se encontra a visualizar os interesses, vai ser apresentado também o *fragment* com o ambiente de pesquisa guardado na *stack*.

Desta forma, a aplicação fica mais leve na transição entre ambientes, não sendo necessário carregamento de *activities*.

Compatibilidade com versões anteriores

Outro aspeto foi a compatibilidade da aplicação com as versões anteriores do *Android*, isto porque elementos como *ActionBar*, *Fragments* e *SearchView* não são suportados por versões do *Android* inferiores à 3.0. Como a aplicação tem o objetivo de suportar desde a versão 2.3.3, esta foi desenvolvida com o auxílio das *Support Libraries*[4] (v4 e v7) do *Android*.

Nas figuras 4.10, 4.11 e 4.12 estão indicados os elementos que foram possíveis de implementar através das *Support Libraries*, permitindo que a aplicação se mantenha coerente nas várias versões.

[3] http://developer.android.com/reference/android/app/Activity.html
[4] https://developer.android.com/tools/support-library/features.html

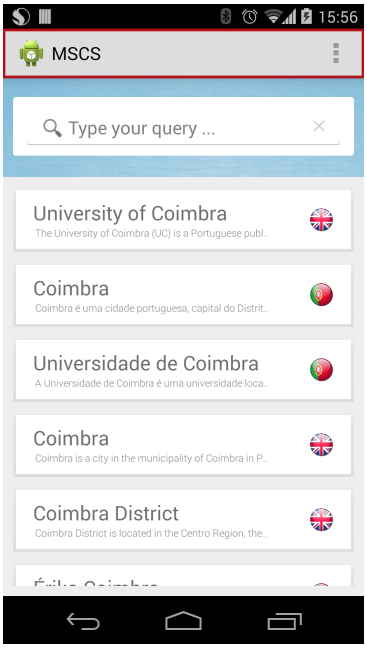


Figura 4.10: *Action Bar* implementada através da *Support Library* v7.

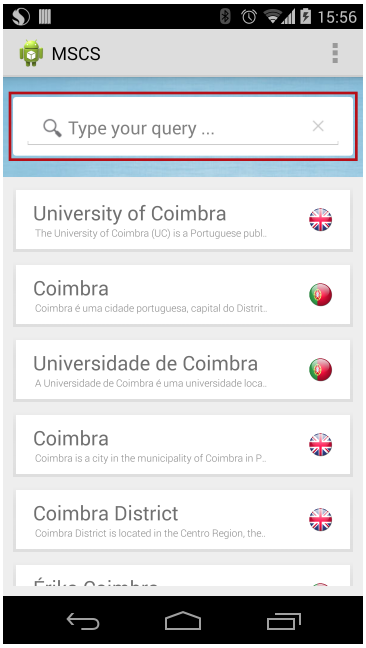


Figura 4.11: *Search View* implementada através da *Support Library* v7.

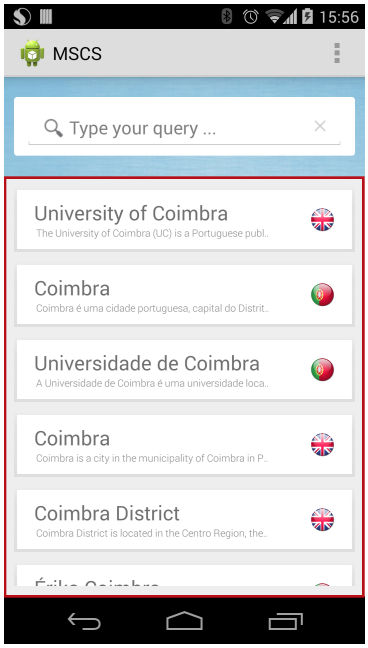


Figura 4.12: *Fragment* implementado através da *Support Library* v4.

4.2 Servidor

No que diz respeito ao servidor, este encontra-se implementado em linguagem *JAVA*. Tal como para a aplicação móvel, é feita uma divisão das funcionalidades do servidor para um melhor entendimento.

4.2.1 Gestão dos Pedidos

Os tipos de pedidos que são recebidos por parte do servidor, assim como os serviços que este tem de executar periodicamente acabam por se tornar um problema. Isto porque é necessário permitir que essas operações possam ser tratadas em simultâneo. São apresentadas na figura 4.13 as operações que o servidor deve realizar.

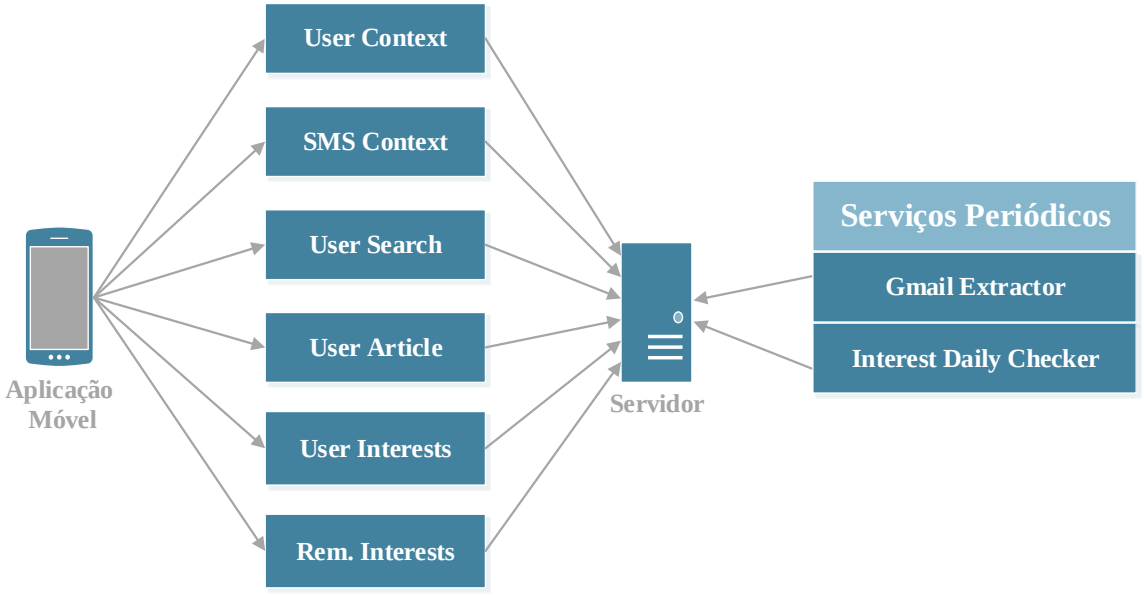


Figura 4.13: Tipo de operações que o servidor suporta.

Relativamente aos pedidos provenientes da aplicação móvel, estes implicam que o servidor realize as operações descritas:

- ***User Context:*** Este pedido é feito quando um utilizador entra na aplicação móvel pela primeira vez, ou seja, é executado quando o utilizador acaba de se registar no sistema de forma a guardar o IMEI do seu dispositivo, assim como as suas credenciais Gmail.
- ***SMS Context:*** Quando é solicitado ao servidor esta operação, este deve proceder a extração de interesses sobre a SMS ou SMS's, para assim os guardar no contexto do utilizador.
- ***User Search:*** Neste pedido, deve ser recolhida a *query* que foi enviada pela aplicação móvel para que seja feita a pesquisa sobre o índice da DBpedia, baseada no contexto do utilizador. Por fim, o servidor deve enviar como resposta a lista de resultados.
- ***User Article:*** O servidor recebe este pedido quando do lado da aplicação móvel o utilizador consulta um artigo. A operação feita pelo servidor é inserir o tópico do artigo nos interesses do utilizador ou caso este já faça parte, atualizar o peso do mesmo.
- ***User Interests:*** Esta funcionalidade requer que o servidor recolha os interesses do utilizador que se encontram armazenados e os envie para a aplicação móvel, de forma a estes poderem ser visualizados.

- ***Remove Interest:*** Aqui é solicitado ao servidor para eliminar um interesse que o utilizador não considerou correto. Para tal, o servidor identifica-o através do URI que é recebido e procede à remoção do mesmo.

Quanto aos serviços periódicos que este tem de executar:

- ***Gmail Extractor:*** O servidor quando a execução deste serviço é solicitado (60 em 60 minutos), deve proceder à extração de interesses sobre os *emails* dos utilizadores, de forma a guardá-los no contexto de cada um.
- ***Interests Daily Checker:*** Este serviço é executado diariamente, a uma hora em que a utilização do servidor é menor, isto é, às 4:00 AM. Tem como objetivo fazer uma atualização dos interesses dos utilizadores de acordo com a última vez que foram modificados.

Como tal, para que o servidor suportasse executar todas estas operações em paralelo, foi necessária a utilização de *Thread Pools* para a realização do mesmo.

Cada tipo de pedido que é solicitado ao servidor tem um *Executor Service* (ES) criado para si. Por sua vez, cada ES tem ao seu dispôr uma *Thread Pool*, sendo que esta é constituída por *Worker Threads* específicas para realizar aquela operação. Para além disso, para cada *Thread Pool* está definida uma *queue* que permite colocar pedidos em espera, caso os *workers* estejam todos ocupados. Desta forma, quando o servidor tem de executar algum pedido, este é encaminhado para a respetiva *queue*, aguardando que um dos *workers* o retire da lista.

Com este mecanismo implementado, a aplicação servidor passa a conseguir executar vários pedidos em paralelo, tornando-se mais eficiente. A figura 4.14 demonstra como se encontra estruturado este mecanismo.

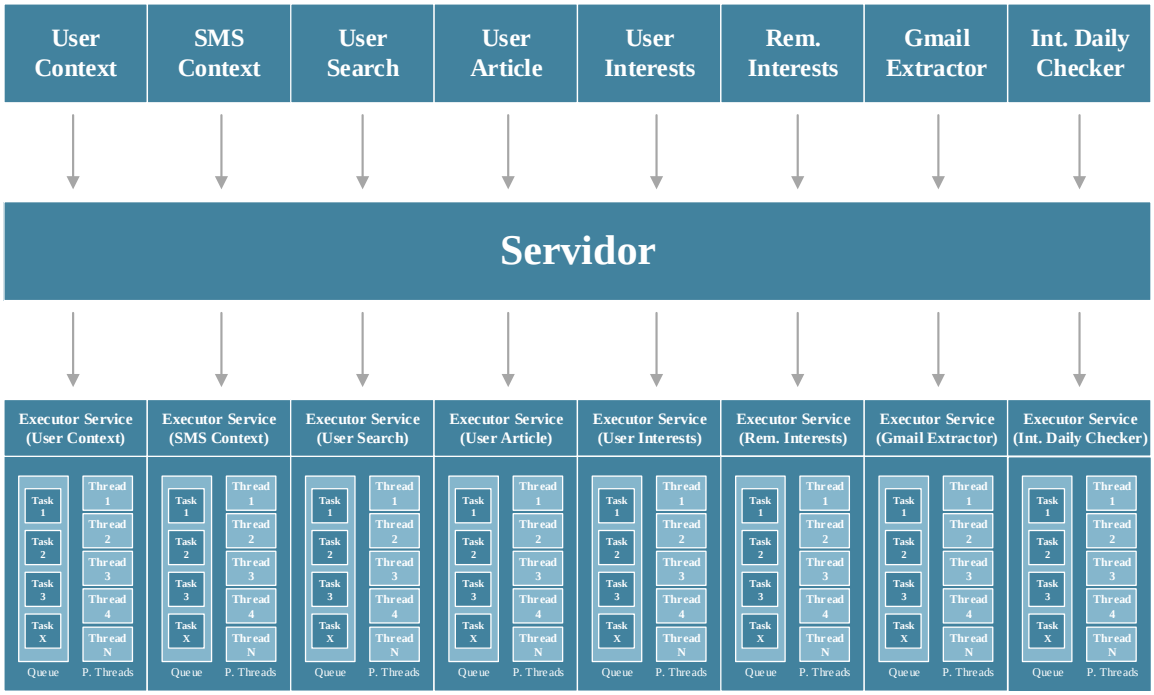


Figura 4.14: Gestão dos pedidos através de *Thread Pools*.

4.2.2 Extração de Interesses

A extração de interesses como já foi referido, é feita tanto nos *emails* recebidos pelo utilizador, como pelas SMS's enviadas e recebidas. O processo é feito para duas línguas, português e inglês, sendo que foi necessário utilizar ferramentas diferentes para conseguir suportar as duas línguas, como será possível verificar mais à frente neste tópico. De forma a entender de uma forma mais clara como é feito o processo, este é explicado por etapas.

Deteção da Linguagem

Como o sistema suporta duas linguagens (português e inglês), é necessário primeiro detetar qual o idioma em que o texto recebido para extração se encontra. Para isso, utiliza-se o módulo *language* do projeto KISWSNLP descrito na subsecção 3.4.2.

Inicialmente é recolhido o texto presente no *email* ou SMS em questão. De seguida a partir da classe *LanguageManager* é verificado o número de *Stopwords* encontradas. Por fim, se existir um maior número destas palavras em português, é considerado que o *email*/SMS encontra-se escrito em português, caso contrário é considerado como idioma a língua inglesa.

Construção dos Interesses

Para a construção de interesses foi necessário implementar um mecanismo, que para além de complexo, fez uso de tecnologias[5] como *Stanford NER* [38] e *Semantic Social Mining NER* [39], para Reconhecimento de Entidades Mencionadas (REM) em inglês e português respetivamente. Para além destas duas, foi também utilizado o *Part-of-Speech* (POS) *Tagging* do OpenNLP[6] que suporta as duas línguas. De seguida, são apresentados alguns detalhes quanto ao uso destas tecnologias.

Relativamente ao REM, são apresentadas na tabela 4.1 os tipos de entidades que podem ser encontradas tanto para a língua inglesa como portuguesa.

Stanford NER	Semantic Social Mining NER	Exemplos
ORGANIZATION	ORGANIZAÇÃO	Mc Donalds, SONAE, etc
LOCATION	LOCAL	Hong-Kong, Porto, etc
PERSON	PESSOA	Steve Jobs, Bill Gates, etc

Tabela 4.1: Tipos de entidades utilizadas para o REM.

Para o OpenNLP POS *Tagger*, são apresentadas algumas das possíveis *tags* na tabela 4.2.

Tags EN	Classe Gramatical	Tags PT	Classe Gramatical
NNP	Proper Noun, singular	PROP	Nome Próprio
NNPS	Proper Noun, plural	N	Nome Comum
NN	Noun, singular	PRP	Preposição
NNS	Noun, plural	ADJ	Adjetivo
JJ	Adjective	ART	Determinante
IN	Preposition	CONJ	Conjunção
DT	Determiner	ADV	Advérbio
VB	Verb	NUM	Número

Tabela 4.2: Exemplos de *tags* para o processo de POS *Tagging*.

Foram também definidas sequências de palavras mais comuns para a língua inglesa, sendo apresentadas pela tabela 4.3.

[5] A sigla NER (Named-Entity Recognition) é o termo em inglês para REM (Reconhecimento de Entidades Mencionadas).

[6] http://opennlp.sourceforge.net/models-1.5/

Patterns Comuns (EN)			
NNPS+IN+NNPS+NNPS	NN+NNP+NN	NN+NN	NN+JJ
NNP+IN+NNP+NNP	NNP+NNP+NNP	NNPS	NN+JJS
NNPS+IN+NNPS	NNPS+NNPS	NNP	NN+JJR
NNP+IN+NNP	NNP+NNP	NNS	NNS+JJS
NNS+NNPS+NNS	NNS+NNS	NN	NNS+JJR

Tabela 4.3: Exemplos de *patterns* comuns da língua inglesa.

Relativamente à língua portuguesa, estas são demonstradas na tabela 4.4.

Patterns Comuns (PT)			
PROP+PRP+PROP+PROP	PROP+PROP	N+ADJ	N
PROP+PRP+PROP	PROP	N+PRP+N	N+N

Tabela 4.4: Exemplos de *patterns* comuns da língua portuguesa.

Os tipos de entidades, as POS *tags* e as sequências de palavras, são fundamentais para perceber o funcionamento da construção de interesses, daí ter sido feita uma introdução a estes tópicos.

De seguida, é apresentada a simulação do algoritmo que se encontra implementado no módulo *interest* do projeto KISWSNLP, com o objetivo de entender como são extraídos interesses.

Passo 1

Inicialmente é aplicado no texto REM e POS *Tagging*. Este processo vai dar origem a uma lista de palavras onde cada uma pode ter associada uma REM *tag* (ver tabela 4.1) e uma POS *tag* (ver tabela 4.2). Na figura 4.15 é possível verificar o processo para um caso real.

Figura 4.15: Passo 1 da construção de interesses.

Passo 2

Ao obter a lista de palavras classificadas, esta vai ser percorrida de forma a serem construídos interesses. Primeiramente é feita à palavra uma verificação com o objetivo de saber se foi

atribuída uma REM *tag* a esta. Neste passo, foi atribuído *No Classification* (NC) à palavra *"Get"*. Desta forma, esta não se trata de um possível interesse e é descartada, tal como se pode verificar na figura 4.16.

Figura 4.16: Passo 2 da construção de interesses.

Passo 3

Passando à próxima palavra da lista, *"Scribd"*, é efetuada a verificação da sua classificação REM. Como a *tag* atribuída é *ORGANIZATION*, esta palavra trata-se de uma entidade, o que valida a verificação.

De seguida, é necessário verificar se *"Scribd"* consiste num padrão válido, isto é, numa palavra ou junção de palavras que formem uma sequência de classes gramaticais comum na língua inglesa neste caso (ver tabela 4.3). Como a sequência é válida (NNP - *Proper Noun Singular*) este possível interesse passa à próxima fase.

Como última verificação, é necessário percorrer a lista de *Stopwords* de forma a saber se o possível interesse está inserido na mesma. No caso de *"Scribd"*, este não pertence a essa lista, sendo então validado e adicionado à lista de interesses (ver figura 4.17).

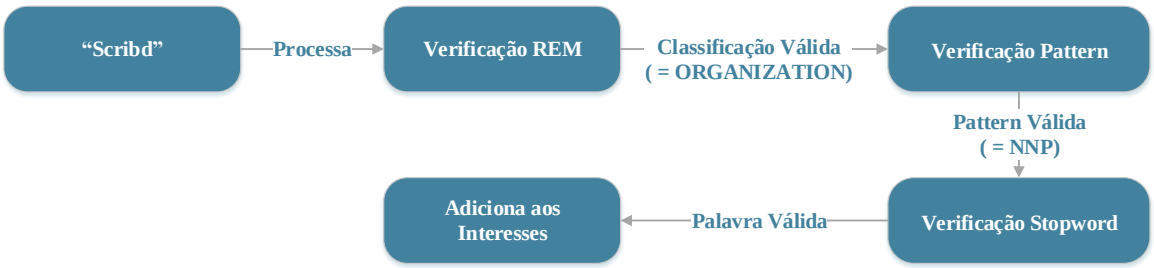


Figura 4.17: Passo 3 da construção de interesses.

Passo 4

Como a palavra anterior se tratava de uma entidade válida, vai ser feita uma junção desta com a palavra seguinte da lista, no sentido de verificar se é possível construir um interesse ainda mais elaborado. Sendo assim, a junção resultou em *"Scribd on"*, que não passou na verificação REM pois a REM *tag* para a palavra *"on"* é NC.

Desta forma, não é possível construir mais interesses com a palavra *"Scribd"*, estando já esta guardada como interesse. A palavra *"on"* é então descartada tal como aconteceu no passo 2 para a palavra *"Get"*. A figura 4.18 demonstra o processo.

Figura 4.18: Passo 4 da construção de interesses.

Passo 5

À semelhança do que aconteceu no passo 3, neste retirou-se a palavra *"Kindle"* da lista e esta passou nas três verificações (***REM Tag:*** *ORGANIZATION* → ***Pattern:*** *NNP* → ¬***Stopword***), sendo então adicionado à lista de interesses (ver figura 4.19).

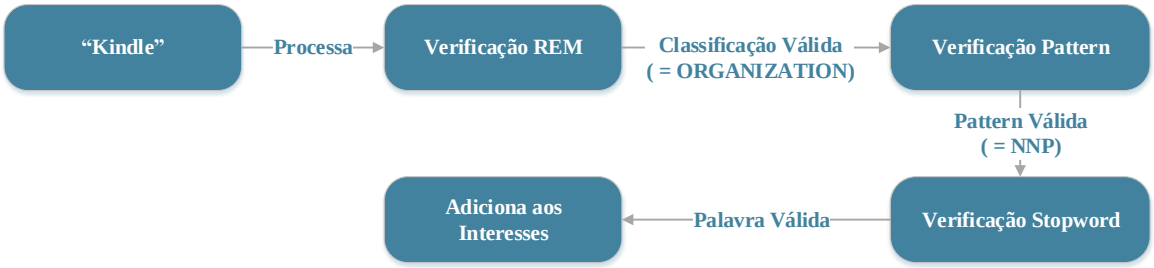


Figura 4.19: Passo 5 da construção de interesses.

Passo 6

Como foi feito para o passo 4, a palavra anterior foi adicionada aos interesses. Assim, vai ser feita a junção com a palavra seguinte no sentido de verificar se é possível construir um novo interesse.

Neste caso, a palavra conjunta passou por todas as verificações (***REM Tag:*** *ORGANIZATION + ORGANIZATION* → ***Pattern:*** *NNP + NNP* → ¬***Stopword***). Desta forma, vai ser adicionado à lista, o interesse *"Kindle Fire"*. Na figura 4.20 está demonstrado como é que foi feita a validação do mesmo.

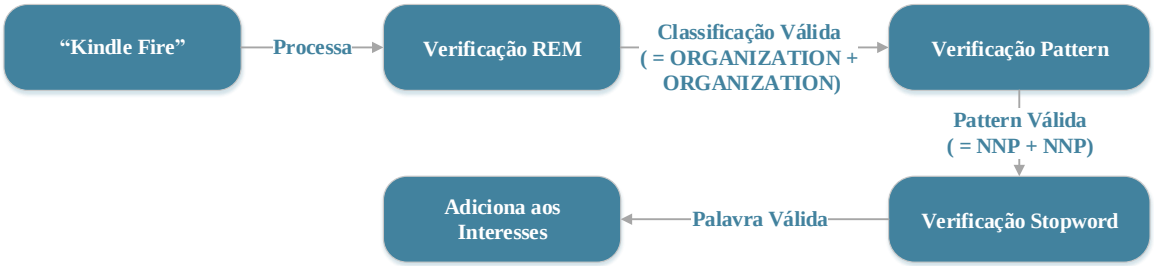


Figura 4.20: Passo 6 da construção de interesses.

Passo 7

No passo 7, tenta-se fazer uma nova junção de palavras, sendo que o resultado foi *"Kindle Fire!"*. Tal como aconteceu nos passos 2 e 4, a verificação REM detetou uma junção de palavras inválida (*ORGANIZATION* + NC), resultando no descarte da palavra como podemos verificar na figura 4.21.

Figura 4.21: Passo 7 da construção de interesses.

Por fim, a lista de palavras classificadas foi percorrida até ao fim, sendo que a lista de interesses resultante deste processo passa para um mecanismo que faz a associação dos interesses ao utilizador.

4.2.3 Associação de Interesses

Quando são construídos interesses a partir de *emails* e SMS's, estes resultam numa lista de interesses que ainda deve ser avaliada de forma a verificar o artigo da DBpedia a que se referem.

No entanto, neste processo, ocorrem problemas de ambiguidade de palavras. Para resolver este problema foi criado um modelo de tópicos para os artigos da DBpedia a partir da API do *Mallet*[7] que permite fazer *Topic Modeling*[8].

Para a criação do modelo, foi percorrido o índice da DBpedia de forma a ser aplicado *Topic Modeling* ao *abstract* de cada um dos artigos indexados (ver figura 4.22).

[7] http://mallet.cs.umass.edu/
[8] http://mallet.cs.umass.edu/topics.php

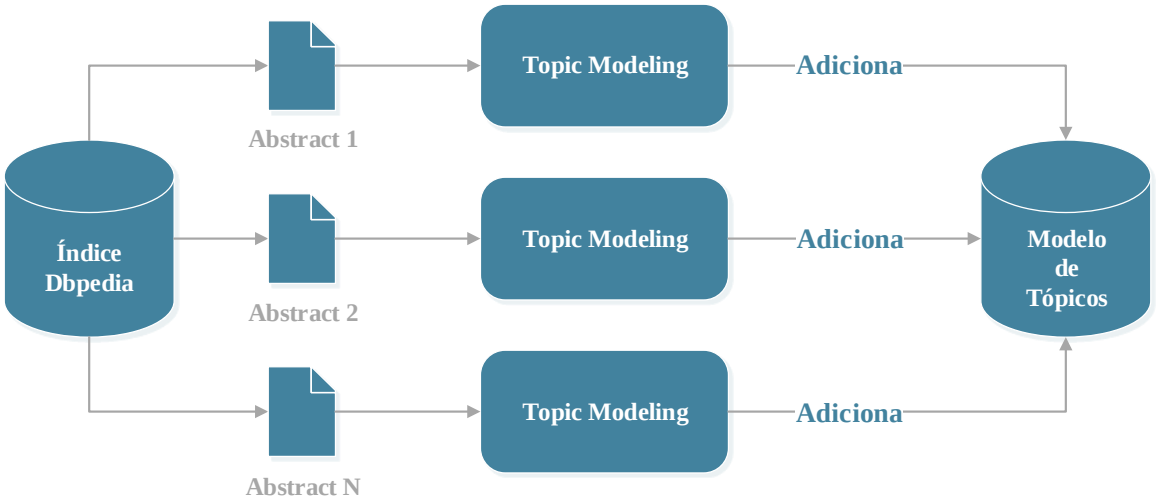


Figura 4.22: Criação do modelo de tópicos a partir dos *abstracts* dos artigos.

De seguida, foi criado um novo índice da DBpedia a partir do repositório da mesma. O objetivo desta operação foi realizar *Topic Inference* do *abstract* de cada um dos artigos da DBpedia sobre o modelo de tópicos criado, obtendo uma lista com graus de pertença[9] do artigo a cada um dos tópicos do modelo.

Cada uma destas listas foram adicionadas como atributos do artigo na nova indexação, permitindo assim que fossem acedidas posteriormente durante a execução do servidor. A figura 4.23 demonstra como é realizado o processo para um dos artigos a indexar.

[9]Grau de Pertença - Especifica o quão se adequa um documento a um tópico, através de um valor de percentagem.

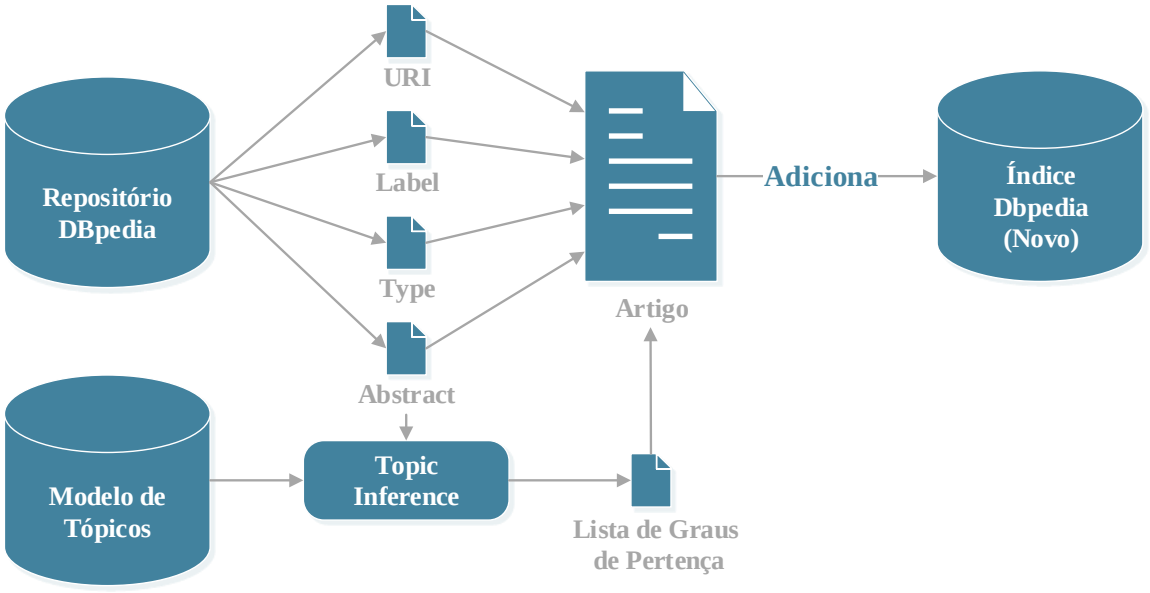


Figura 4.23: Criação do novo índice da DBpedia com a lista de graus de pertença.

Com o modelo de tópicos criado e os artigos da DBpedia devidamente indexados, isto é, com a lista de graus de pertença aos tópicos, é possível desambiguar a associação de um novo interesse do utilizador.

Para isso, quando se inicia a associação de interesses a partir da lista construída na subsecção 4.2.2, os elementos desta são inferidos no modelo de tópicos criado, resultando numa lista com graus de pertença da SMS ou *email* a cada um dos tópicos do mesmo (ver figura 3.4).

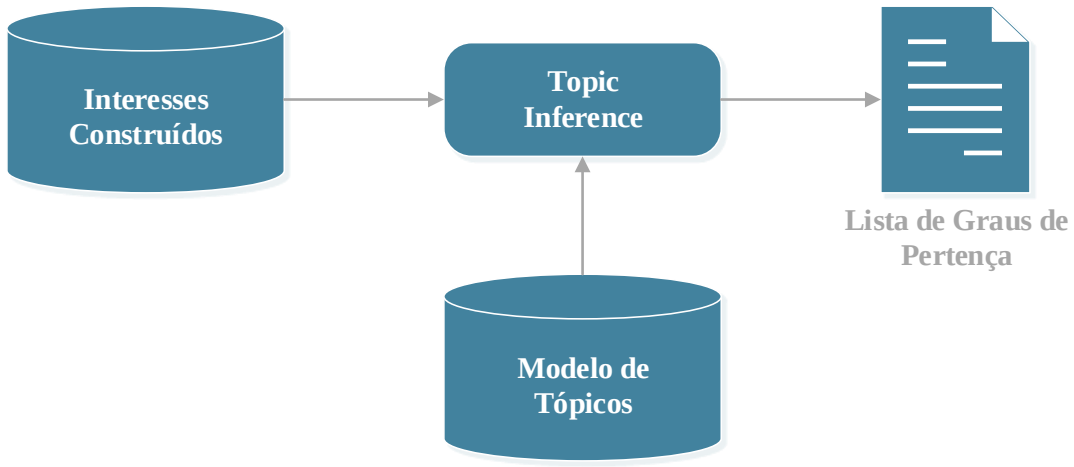


Figura 4.24: Criação da lista de graus de pertença referente aos interesses construídos.

De seguida, cada interesse da lista vai ser procurado no índice dos artigos da DBpedia. Para cada

um, vai ser devolvida uma lista de resultados, isto é, possíveis artigos que possam ser associados, sendo que estes foram selecionados de acordo com a ocorrência do interesse em cada um. É possível verificar através figura 4.25 o procedimento.

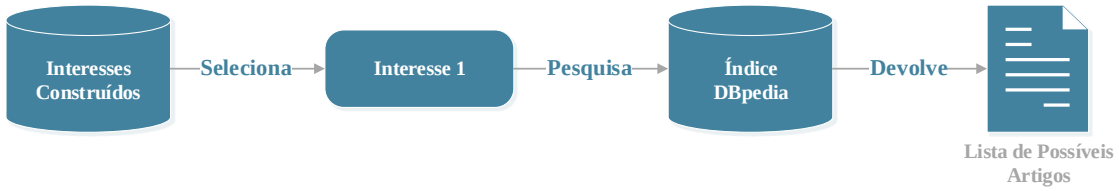


Figura 4.25: Representação do processo de pesquisa do interesse no índice da DBpedia.

Como a ocorrência do interesse nem sempre é suficiente para saber qual o artigo correto, são utilizados os tópicos do modelo criado para desambiguar o interesse. Para isso, cada um dos artigos vai ser comparado com o interesse através das respetivas listas de graus de pertença. É feita uma similaridade de co-senos entre as duas listas e a que obtiver maior valor é correspondente ao artigo que deve ser associado ao interesse. Na figura 4.26 é possível verificar o método de desambiguação.

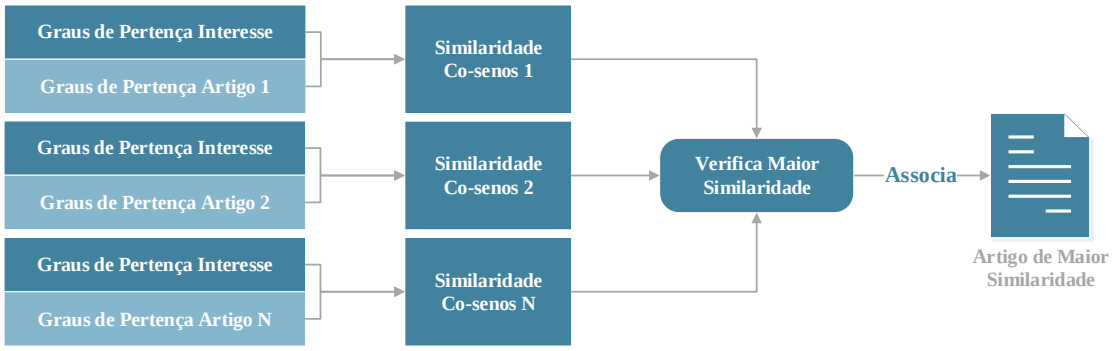


Figura 4.26: Diagrama representativo do processo de desambiguação do interesse.

4.2.4 Interpretação da Pesquisa

Quando é solicitado ao servidor processar uma pesquisa efetuada na aplicação móvel, a primeira etapa dá-se pela interpretação da *query*, pois esta é procurada no índice dos artigos da DBpedia devolvendo uma lista de possíveis resultados. Estes são selecionados através do número de ocorrências da *query* no artigo.

À medida que esta lista é preenchida, vai sendo também feito para cada resultado uma verificação se este já está inserido nos interesses do utilizador. O objetivo deste processo é atribuir a cada um

dos resultados, caso estes façam parte do contexto em questão, o peso que este tem nos interesses do utilizador. Assim, quando a lista de resultados da pesquisa for apresentada ao utilizador, esta encontra-se ordenada do mais significativo para o menos significativo. A figura 4.27 permite clarificar o processo de pesquisa.

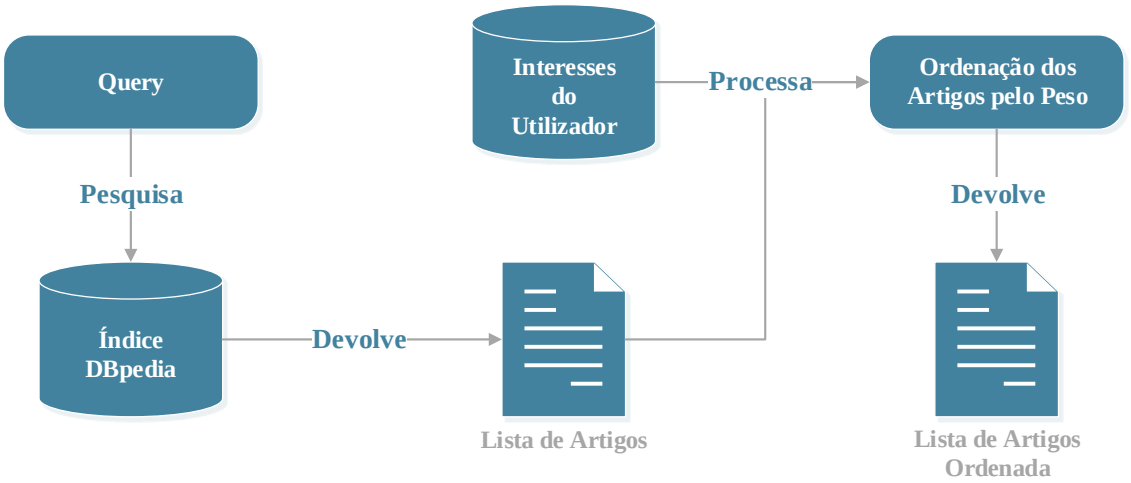


Figura 4.27: Funcionamento do processo de pesquisa do lado do servidor.

4.2.5 Evolução dos Interesses

Os interesses do utilizador que se encontram armazenados são fundamentais no contexto do mesmo, isto porque os pesos que se encontram atribuídos a eles, definem o que é mais importante para o utilizador naquele momento.

No entanto, o utilizador não mantém sempre a mesma ordem de preferência dos seus interesses, isto é, numa semana pode estar mais focado em assuntos tecnológicos (ex: Google Nexus, JAVA, Raspberry Pi, etc) e na semana seguinte ter de lidar com temas desportivos (ex: Champions League, NBA, etc).

Para lidar com estas transições de temas, foi implementado um mecanismo que permite ajustar os pesos dos interesses do utilizador face à atualidade.

Inicialmente, quando um interesse novo é adicionado ao contexto do utilizador, este é inserido com o peso igual a 0.1. O facto do valor inicial ser 0.1, indica que o interesse tem um significado mínimo para o utilizador, visto que a escala é entre 0.1 e 1 (0.1 - Pouco Relevante / 1 - Muito Relevante).

De forma a obter um incremento da importância do interesse prolongada, isto é, não atingir o valor 1 rapidamente, optou-se por definir um limite diário de incrementação da importância do

interesse (0.2). Para além disso, cada vez que o interesse é acedido durante esse dia, o valor de incremento também vai diminuindo, sendo a fórmula deste:

$$Inc = LimiteDiario/2^{Num.Acessos} \tag{4.1}$$

Na figura 4.28 é possível ver o funcionamento deste mecanismo sobre um interesse novo no contexto do utilizador.

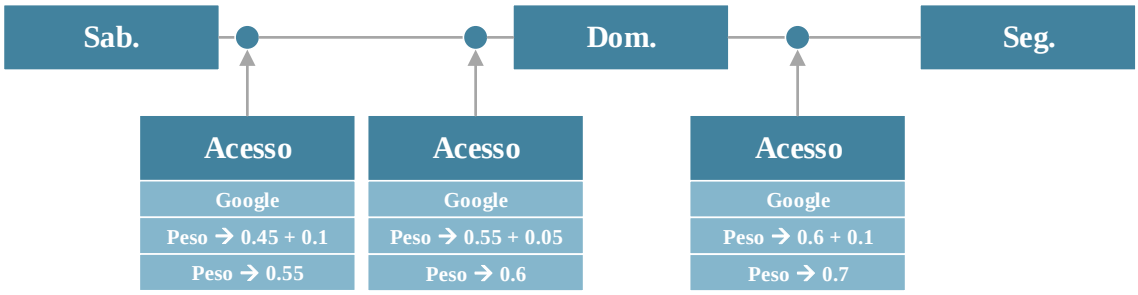


Figura 4.28: Incrementação do peso do interesse ao longo dos dias.

Relativamente à importância dos interesses decrementar, encontra-se implementado um serviço diário (*Interest Daily Checker*), referenciado na subsecção 4.2.1, que faz essa gestão.

Este mecanismo verifica diariamente (4:00 AM) se houve interação com o interesse, isto é, se houve uma atualização do peso do mesmo. Caso este tenha ficado 24 horas ou mais sem sofrer qualquer alteração, deve ser decrementado do seu peso 0.05 (metade do valor mínimo de incrementação). Durante este processo, se o peso do interesse atingir um valor menor que 0.1 (valor mínimo), então este é removido do contexto, pois naquela fase, não tem importância para o utilizador. É apresentado na figura 4.29 o funcionamento do serviço *Interest Daily Checker* para um interesse presente no contexto do utilizador que tinha sido modificado recentemente.

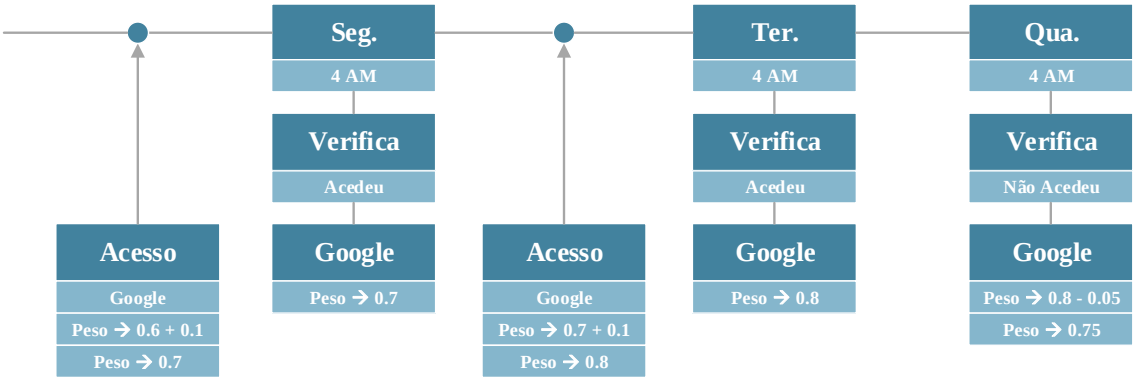


Figura 4.29: Representação do mecanismo de decrementação do peso de interesses.

Capítulo 5

Experimentação

Após o desenvolvimento do sistema *Mobile Semantic Context-Based Search* (MSCS) foi necessário passar à realização de experiências que permitissem avaliar o funcionamento do mesmo. Para isso, o processo foi dividido em duas fases, a primeira que teve como objetivo recolher informação do utilizador, no sentido de ser possível construir contexto para os vários utilizadores. A segunda etapa passou por uma sessão de pesquisas com os utilizadores, de forma a avaliar a influência do contexto nos resultados. O objetivo desta experimentação passa por atingir resultados positivos, isto é, demonstrar que o contexto influencia positivamente os resultados das pesquisas.

5.1 Recolha de Informação

Para a realização da primeira etapa foi necessário instalar a aplicação móvel em dispositivos de utilizadores reais, isto porque o objetivo do sistema é construir contexto através de informação proveniente dos mesmos.

Desta forma, foram conseguidos 6 utilizadores de teste, sendo que estes têm experiência elevada no que diz respeito a pesquisas em motores de busca.

Após a instalação da aplicação nos dispositivos dos utilizadores, foi feita extração de interesses a partir das suas SMS's e *emails* durante 15 dias. Na tabela 5.1 encontram-se apresentados para cada utilizador, o número de interesses extraídos, o número de interações (SMS's e *emails* que resultaram em interesses), assim como a versão *Android* do dispositivo, no sentido de comprovar a compatibilidade de versões referida na subsecção 4.1.5.

Utilizador	Versão Android	Num. Interações	Num. Interesses
USER1	4.4	87	223
USER2	4.1	91	193
USER3	4.3	82	275
USER4	4.2	53	167
USER5	2.3.3	77	217
USER6	4.0	102	317

Tabela 5.1: Resultados da recolha de informação dos utilizadores.

5.2 Sessão de Pesquisas

Após terem sido analisados durante duas semanas *emails* e SMS's de cada utilizador, foi feita uma avaliação ao sistema no sentido de verificar o quão influente se tornou a construção de contexto do utilizador. Esta avaliação passou por uma sessão de pesquisas com cada um, sendo que os resultados devolvidos viriam ordenados pela importância para o utilizador.

A cada pesquisa feita, o utilizador consultou os artigos que eram do seu interesse, e o que se pretendeu desta operação foi os artigos selecionados serem avaliados em duas listas de resultados, com e sem contexto do utilizador. Ao comparar o artigo selecionado nestas duas listas, pretendia-se que este se encontrasse o mais para cima possível na lista com contexto e o mais para baixo na sem contexto. A figura 5.1 torna mais perceptível o que se pretendeu desta consulta de artigos.

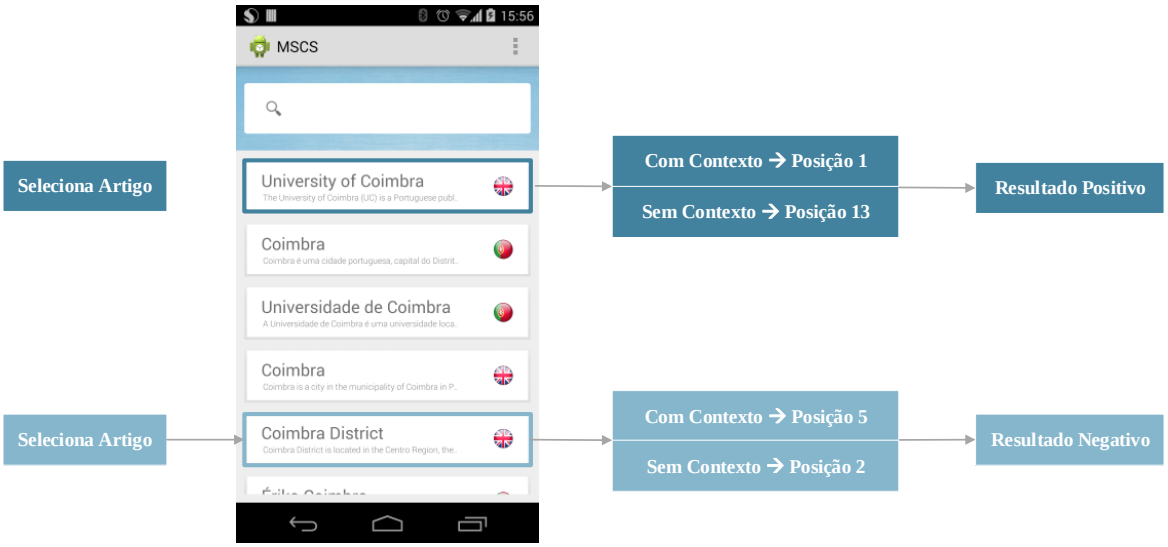


Figura 5.1: Representação do resultado positivo e negativo da consulta de artigos.

Como se pode verificar, existem dois tipos de resultados possíveis: positivo e negativo. O resultado positivo significa que o utilizador quando selecionou o artigo na lista com contexto, este encontrava-se numa posição mais favorável do que numa lista sem contexto, isto é, mais para cima. Caso o resultado seja negativo, ou seja, o artigo pretendido na lista de contexto estar numa posição mais a baixo do que na sem contexto, então o utilizador não se encontra a tirar proveito nenhum do contexto.

Na sessão feita, foram realizadas 20 pesquisas por cada utilizador. Desta forma, foi possível retirar resultados sobre a influência do contexto nas pesquisas dos mesmos. Na tabela 5.2, estão presentes para cada utilizador, o número de interesses, o número de consultas de artigos e por fim a percentagem de resultados positivos na consulta de artigos. Desta forma, é possível analisar o impacto do contexto nas pesquisas.

Utilizador	Num. Interesses	Num. Consultas	Resultados Positivos
USER1	223	39	56%
USER2	193	29	48%
USER3	275	31	61%
USER4	167	28	46%
USER5	217	28	54%
USER6	317	34	59%

Tabela 5.2: Resultados da sessão de pesquisas dos utilizadores.

A partir destes resultados, é possível verificar que o contexto construído para cada utilizador é significativo na apresentação dos artigos da DBpedia ao mesmo. Esta avaliação é feita a partir da percentagem de resultados que foram consultados e que foram classificados como positivos. Apesar de dois utilizadores atingirem uma percentagem menor do que 50%, são valores muito próximos do mesmo e que indicam que o contexto influencia os resultados apresentados.

Outra aspecto que se pode salientar, é o facto de que para as sessões realizadas, o número de interesses recolhidos faz a diferença. Isso pode ser verificado através da diferença de interesses recolhidos para os utilizadores 2 e 4. Estes obtiveram um número de interesses recolhidos menor que os restantes, e o número de resultados positivos demonstrou-se inferior em relação aos restantes utilizadores.

No entanto, pode-se concluir também, que recolher muitos interesses nem sempre resulta em mais resultados positivos. Se analisarmos os resultados do utilizador 3 e 6, que são os que obtiveram melhores resultados, verifica-se que o utilizador 3 teve melhor percentagem com um menor número de interesses recolhidos e com um menor número de consultas de artigos. Uma

das causas destas percentagens pode ser o facto de os interesses do utilizador 6 estarem mais dispersos, ou seja, abordarem múltiplos temas e estes não terem sido pesquisados. Ao contrário do utilizador 3 que teria os seus interesses mais à volta de um tema e quando foram efetuadas as pesquisas, foram consultados vários artigos que estariam dentro deste tema, resultando em mais consultas positivas.

Como tal, os resultados apresentados revelam-se positivos, sendo que em média, foram atingidos 54% dos resultados positivos.

Capítulo 6

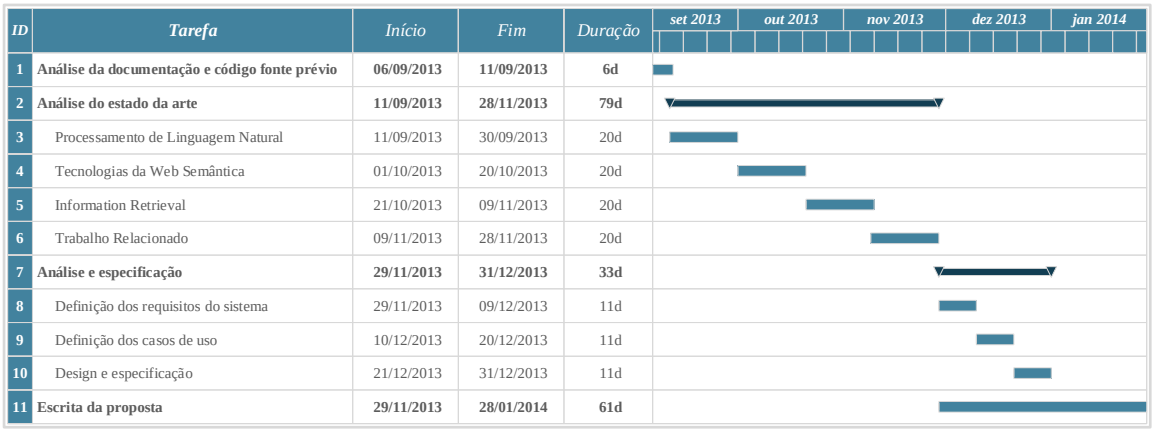
Planeamento

Neste capítulo encontram-se especificadas as várias etapas da realização desta dissertação divididas por semestre. O objetivo é analisar aquilo que foi realizado durante cada semestre, assim como as contrariedades que surgiram no prolongar do mesmos.

6.1 Primeiro Semestre

Previamente ao desenvolvimento do sistema, foi necessário proceder a um estudo prévio sobre como se encontravam os vários processos de Processamento de Linguagem Natural (PLN), Web Semântica (WS) e *Information Retrieval* (IR), num ambiente de interpretação do contexto de uma pessoa.

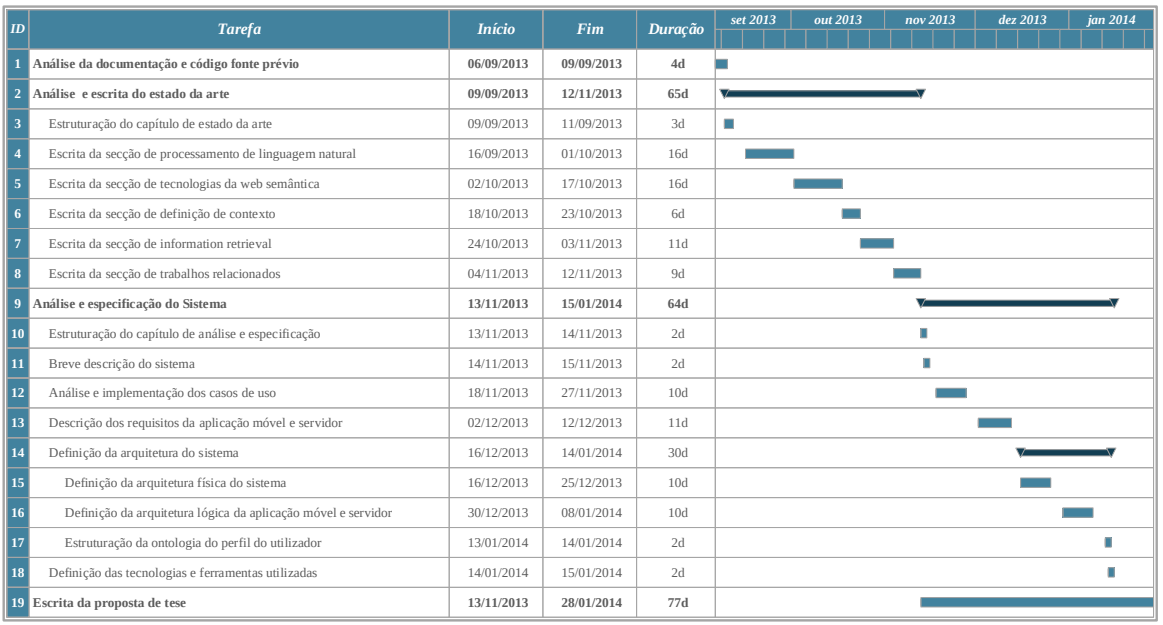
Posteriormente, foram construídos os cenários que permitiram o desenvolvimento deste sistema, desde os tipos de interação que o utilizador teria com o sistema (casos de uso), as funcionalidades que os dois sistemas (aplicação móvel e servidor) deveriam suportar e que tipo de ações devem tomar, assim como estrutura destes. O seguinte diagrama de *Gantt* visa representar, então o plano inicialmente definido de uma forma geral (ver figura 6.1).



ID	Tarefa	Início	Fim	Duração
1	**Análise da documentação e código fonte prévio**	**06/09/2013**	**11/09/2013**	**6d**
2	**Análise do estado da arte**	**11/09/2013**	**28/11/2013**	**79d**
3	Processamento de Linguagem Natural	11/09/2013	30/09/2013	20d
4	Tecnologias da Web Semântica	01/10/2013	20/10/2013	20d
5	Information Retrieval	21/10/2013	09/11/2013	20d
6	Trabalho Relacionado	09/11/2013	28/11/2013	20d
7	**Análise e especificação**	**29/11/2013**	**31/12/2013**	**33d**
8	Definição dos requisitos do sistema	29/11/2013	09/12/2013	11d
9	Definição dos casos de uso	10/12/2013	20/12/2013	11d
10	Design e especificação	21/12/2013	31/12/2013	11d
11	**Escrita da proposta**	**29/11/2013**	**28/01/2014**	**61d**

Figura 6.1: Planeamento inicial para as tarefas realizadas no primeiro semestre.

Como tal, este era o plano pensado inicialmente para o desenvolvimento desta dissertação, mas que no entanto sofreu algumas alterações, mais precisamente, ao nível dos intervalos temporais. A figura 6.2, demonstra as alterações que o processo de desenvolvimento sofreu, assim como as tarefas concretizadas ao longo do tempo de uma forma mais detalhada.



ID	Tarefa	Início	Fim	Duração
1	**Análise da documentação e código fonte prévio**	**06/09/2013**	**09/09/2013**	**4d**
2	**Análise e escrita do estado da arte**	**09/09/2013**	**12/11/2013**	**65d**
3	Estruturação do capítulo de estado da arte	09/09/2013	11/09/2013	3d
4	Escrita da secção de processamento de linguagem natural	16/09/2013	01/10/2013	16d
5	Escrita da secção de tecnologias da web semântica	02/10/2013	17/10/2013	16d
6	Escrita da secção de definição de contexto	18/10/2013	23/10/2013	6d
7	Escrita da secção de information retrieval	24/10/2013	03/11/2013	11d
8	Escrita da secção de trabalhos relacionados	04/11/2013	12/11/2013	9d
9	**Análise e especificação do Sistema**	**13/11/2013**	**15/01/2014**	**64d**
10	Estruturação do capítulo de análise e especificação	13/11/2013	14/11/2013	2d
11	Breve descrição do sistema	14/11/2013	15/11/2013	2d
12	Análise e implementação dos casos de uso	18/11/2013	27/11/2013	10d
13	Descrição dos requisitos da aplicação móvel e servidor	02/12/2013	12/12/2013	11d
14	Definição da arquitetura do sistema	16/12/2013	14/01/2014	30d
15	Definição da arquitetura física do sistema	16/12/2013	25/12/2013	10d
16	Definição da arquitetura lógica da aplicação móvel e servidor	30/12/2013	08/01/2014	10d
17	Estruturação da ontologia do perfil do utilizador	13/01/2014	14/01/2014	2d
18	Definição das tecnologias e ferramentas utilizadas	14/01/2014	15/01/2014	2d
19	**Escrita da proposta de tese**	**13/11/2013**	**28/01/2014**	**77d**

Figura 6.2: Planeamento cumprido para as tarefas realizadas no primeiro semestre.

Começando por referenciar a análise e escrita do estado da arte, foi conseguida a conclusão de uma primeira versão antes da data estipulada pelo plano inicial. No entanto, esta foi revista pelo professor orientador desta dissertação e assim teve de sofrer algumas alterações quer de estrutura, quer de algum conteúdo, sendo fechada por completo numa fase mais tardia, mas que não comprometeu o desenvolvimento da restante dissertação.

A análise e especificação do sistema já marcou alguma diferença em relação ao planeado inicialmente, principalmente no que diz respeito à janela temporal definida para o mesmo. Esta prolongou-se até Janeiro, sendo que inicialmente, abrangia apenas o mês de Dezembro. No entanto, esta alteração aconteceu devido à grande quantidade de especificações que eram necessárias definir relativamente à arquitetura do sistema, ou seja, da aplicação móvel e do servidor. Os requisitos de cada um destes componentes também necessitaram de uma grande atenção para garantir o bom funcionamento do sistema.

De uma forma geral pode-se verificar que o planeamento que foi cumprido segue muitas das vezes uma abordagem sequencial, isto é, para avançar para uma próxima tarefa era necessário já ter acabado outras, como é o caso por exemplo da análise e especificação do sistema, que implicava já ter sido terminada a escrita do estado da arte. No entanto, se nos referirmos à escrita da proposta de tese, esta já seguiu precisamente o contrário, tendo sido começada após a entrega da primeira versão do estado da arte. Esta medida tornou-se crucial permitindo que não fossem corridos riscos devido à extensão de tempo que ocorreu durante a definição da arquitetura.

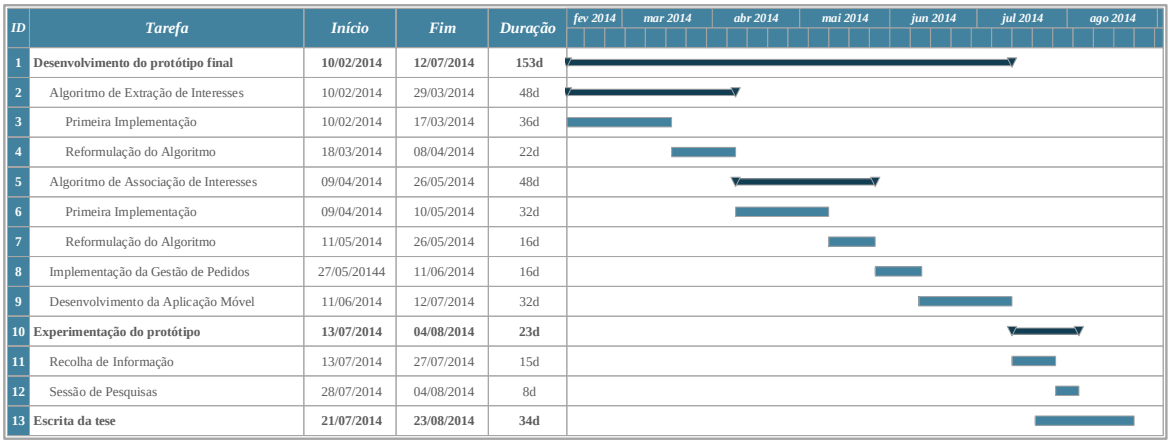
6.2 Segundo Semestre

Visto que o primeiro semestre, foi mais orientado para os conceitos que deveriam ser compreendidos para o desenvolvimento do protótipo final do sistema *Mobile Semantic Context-Based Search*, no segundo semestre passou-se ao desenvolvimento de um protótipo final. Na figura 6.3 está demonstrado plano de trabalho inicial que se pretendia.

ID	Tarefa	Início	Fim	Duração
1	**Desenvolvimento do protótipo final**	**10/02/2014**	**15/05/2014**	**95d**
2	Algoritmo de Extração de Interesses	10/02/2014	13/03/2014	32d
3	Algoritmo de Associação de Interesses	14/03/2014	14/04/2014	32d
4	Desenvolvimento da Aplicação Móvel	15/04/2014	15/05/2014	31d
5	**Experimentação do protótipo**	**16/05/2014**	**15/06/2014**	**31d**
6	**Escrita da tese**	**02/06/2014**	**02/07/2014**	**31d**

Figura 6.3: Planeamento inicial para as tarefas realizadas no segundo semestre.

Tal como aconteceu no primeiro semestre, o plano do segundo semestre também sofreu várias alterações, isto porque surgiram problemas na implementação de algumas funcionalidades e também foram adicionadas novas. O plano demonstrado pela figura 6.4 tem especificado as alterações que foram feitas ao plano.



ID	Tarefa	Início	Fim	Duração
1	**Desenvolvimento do protótipo final**	**10/02/2014**	**12/07/2014**	**153d**
2	Algoritmo de Extração de Interesses	10/02/2014	29/03/2014	48d
3	Primeira Implementação	10/02/2014	17/03/2014	36d
4	Reformulação do Algoritmo	18/03/2014	08/04/2014	22d
5	Algoritmo de Associação de Interesses	09/04/2014	26/05/2014	48d
6	Primeira Implementação	09/04/2014	10/05/2014	32d
7	Reformulação do Algoritmo	11/05/2014	26/05/2014	16d
8	Implementação da Gestão de Pedidos	27/05/20144	11/06/2014	16d
9	Desenvolvimento da Aplicação Móvel	11/06/2014	12/07/2014	32d
10	**Experimentação do protótipo**	**13/07/2014**	**04/08/2014**	**23d**
11	Recolha de Informação	13/07/2014	27/07/2014	15d
12	Sessão de Pesquisas	28/07/2014	04/08/2014	8d
13	**Escrita da tese**	**21/07/2014**	**23/08/2014**	**34d**

Figura 6.4: Planeamento cumprido para as tarefas realizadas no segundo semestre.

Como se pode verificar, a diferença dos dois planos é grande, visto que as tarefas realizadas prolongaram-se por um intervalo de tempo maior.

No desenvolvimento do protótipo final, o algoritmo de extração de interesses, que era suposto estar finalizado em Março, foi apenas fechado em Abril. A causa deste atraso, está relacionada com os resultados que estariam a ser obtidos, ou seja, os interesses extraídos não estavam a ir ao encontro do que se pretendia. Desta forma, foi necessário fazer algumas modificações ao algoritmo com o intuito de obter resultados satisfatórios.

O mesmo foi realizado para a associação de interesses, isto é, depois de uma primeira implementação foi necessário realizar ajustes de forma a melhorar os resultados. Como tal, esta tarefa acabou por apenas ser finalizada em Maio.

Para além destes dois algoritmos, foi necessário proceder à implementação de um mecanismo que tratasse dos pedidos feitos ao servidor (Gestão de Pedidos), de forma a assegurar que este conseguiria atender todos. Esta foi uma tarefa que surgiu durante o desenvolvimento do sistema e que era fundamental para o bom funcionamento da aplicação.

Como tal, a aplicação móvel acabou por passar a ser desenvolvida apenas a partir de Junho, ao contrário do que era estipulado. No entanto, a duração de desenvolvimento da mesma aproximou-se do que se encontrava definido inicialmente.

Relativamente à experimentação do protótipo, esta acabou por ser dividida em duas fases, que já foram explicadas no capítulo 5, acabando por demorar menos tempo do que era estipulado.

Por fim, a escrita da tese foi iniciada durante a experimentação do protótipo, visto que era necessário aguardar que fosse recolhida informação dos dispositivos. Quanto à sua duração, esta foi ao encontro do que estava no plano inicial.

Capítulo 7

Conclusão

Este trabalho surge com o objetivo de melhorar a experiência de pesquisa dos utilizadores em motores de busca. Para isso, foi utilizado no desenvolvimento deste sistema, informação do quotidiano do utilizador, no sentido de construir o seu perfil contextual.

Para obter esta informação do utilizador, foi utilizado o *smartphone* do mesmo para a recolha de SMS's e credenciais do *email*. Para processar a informação recolhida, foi implementado um mecanismo que através de Reconhecimento de Entidades Mencionadas (REM) e *Part-of-Speech* (POS) *Tagging*, construiu interesses do utilizador. Uma vez construídos, estes eram associados a artigos da *Wikipedia*, que se encontravam indexados a partir de um repositório da DBpedia.

Relativamente à gestão dos interesses no perfil do utilizador estes eram diariamente monitorizados, no sentido de limpar os menos utilizados. Analisando os resultados obtidos na experimentação, pode-se dizer que estes são positivos e que demonstram que é possível através de informação recolhida do *smartphone* construir um perfil do utilizador, visto que durante as suas pesquisas, foram obtidos em média mais de 50% de casos positivos.

Desta forma, uma vez realizado o trabalho este contribuiu com:

- Uma aplicação móvel para *Android* capaz de recolher SMS's e credenciais do utilizador
- Um mecanismo de construção de interesses a partir de REM e POS *Tagging* para a língua inglesa e portuguesa
- Um mecanismo de associação de interesses a artigos da DBpedia para a língua inglesa e portuguesa

- Um mecanismo de desambiguação entre interesses e artigos da DBpedia, através de *Topic Modeling*
- Uma ontologia representativa dos interesses de um utilizador
- Um índice de artigos da DBpedia para a língua portuguesa e inglesa
- Um mecanismo de gestão de pedidos entre cliente e servidor
- Um mecanismo de monitorização de interesses do utilizador, para manter estes atualizados

No entanto, algumas limitações são encontradas durante a implementação deste sistema. Ao serem construídos interesses, o algoritmo faz uso de REM, o que por vezes extrai interesses que nada vão significar para o utilizador num ambiente de artigos da DBpedia.

Trabalho Futuro

Apesar de terem sido obtidos resultados positivos, o sistema desenvolvido tem muitas formas de se tornar melhor. Assim, são apresentadas algumas abordagens que se podem seguir:

- Restringir, ainda mais, a construção de interesses, para evitar a criação de alguns muito vagos. Desta forma, é possível tornar ainda mais específico o contexto do utilizador.
- Identificar mais padrões de palavras das línguas portuguesa e inglesa, no sentido de conseguir construir mais interesses.
- Na associação de interesses aos artigos da DBpedia, pode ser melhorado o método de desambiguação.
- Relativamente à extração de interesses das SMS's, é necessário melhorar o método de extração, pois estas mensagens, por vezes, não se encontram estruturadas devidamente (falta de pontuação, abreviaturas, etc), o que dificulta a análise do texto.
- Categorizar os interesses extraídos (Desporto, Tecnologia, etc), com o objetivo de melhorar o processo de desambiguação.

Para além de sugestões relativamente ao que foi implementado, é necessário para trabalho futuro ter consideração por outros aspetos, que não faziam parte do foco principal da dissertação:

- **Privacidade dos dados:** Ao termos acesso a SMS's e *emails* dos utilizadores, estes devem ter a garantia que os seus dados se encontram bem protegidos.

- **Consumo de bateria:** A aplicação móvel ao ter serviços em *background* a serem executados, acaba por ter um acréscimo de consumo de bateria. Desta forma, deve ser aprimorado este processo de forma a reduzir o consumo.
- **Performance do sistema:** No futuro, como um protótipo destes pode dar origem a uma aplicação utilizada por um grande número de utilizadores, deve-se ter cuidados com a sobrecarga que o sistema possa ter.

Bibliografia

[1] Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition (Prentice Hall Series in Artificial Intelligence).* Prentice Hall, 1 edition, 2000. neue Auflage kommt im Fruhjahr 2008.

[2] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific American*, 284(5):34–43, May 2001.

[3] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval.* Cambridge University Press, New York, NY, USA, 2008.

[4] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach (2nd Edition).* Prentice Hall, December 2002.

[5] Noam Chomsky. Three models for the description of language. *IRE Transactions on Information Theory*, 2:113–124, 1956. `http://www.chomsky.info/articles/195609--.pdf` – last visited 14^{th} January 2009.

[6] Raymond Merrill Smullyan. *First-order logic.* Dover publications, New York, 1995.

[7] Anastasia Analyti, Nicolas Spyratos, and Panos Constantopoulos. On the definition of semantic network semantics, 1997.

[8] Charles J. Fillmore. *Frame semantics*, pages 111–137. Hanshin Publishing Co., Seoul, South Korea, 1982.

[9] Nancy Ide and Jean Veronis. Word sense disambiguation: The state of the art. *Computational Linguistics*, 24:1–40, 1998.

[10] Ora Lassila, Ralph R. Swick, World Wide, and Web Consortium. Resource description framework (rdf) model and syntax specification, 1998.

[11] Dan Brickley and R.V. Guha. RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation, World Wide Web Consortium, February 2004.

[12] Sean Bechhofer, Frank Van Harmelen, Jim Hendler, Ian Horrocks, Deborah L McGuinness, Peter F Patel-Schneider, and Lynn Andrea Stein. Owl web ontology language reference. *W3C Recommendation*, 10, 2004.

[13] Eric Prud'hommeaux and Andy Seaborne. Sparql query language for rdf. Latest version available as `http://www.w3.org/TR/rdf-sparql-query/`, January 2008.

[14] Lee Feigenbaum. Semantic web vs. semantic technologies. `http://www.cambridgesemantics.com/semantic-university/semantic-web-vs-semantic-technologies`, April 2012.

[15] G. L. Zuniga. Ontology: its transformation from philosophy to information systems. In *Proceedings of the International Conference on Formal Ontology in Information Systems 2001*, pages 187–197. ACM Press, 2001.

[16] John Sowa. *Knowledge Representation: Logical, Philosophical and Computational Foundations.* Brooks/Cole, Pacific Grove, CA, 2000.

[17] Ian Sommerville. *Software Engineering.* Addison-Wesley, Harlow, England, 9 edition, 2010.

[18] Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowl. Acquis.*, 5(2):199–220, June 1993.

[19] Deborah L. McGuinness and Frank van Harmelen. Owl web ontology language overview. Technical Report REC-owl-features-20040210, W3C, 2004.

[20] N. Guarino. *Formal Ontology in Information Systems: Proceedings of the 1st International Conference June 6-8, 1998, Trento, Italy.* IOS Press, Amsterdam, The Netherlands, The Netherlands, 1st edition, 1998.

[21] G. van Heijst, A. Th. Schreiber, and B. J. Wielinga. Using explicit ontologies in kbs development. *Int. J. Hum.-Comput. Stud.*, 46(2-3):183–292, March 1997.

[22] Natalya F. Noy and Deborah L. Mcguinness. Ontology development 101: A guide to creating your first ontology. Online, 2001.

[23] T. Tran and P. Mika. A survey of semantic search approaches. `http://www.cambridgesemantics.com/semantic-university/semantic-web-vs-semantic-technologies`, 2012.

[24] George A. Miller. Wordnet: A lexical database for english. *COMMUNICATIONS OF THE ACM*, 38:39–41, 1995.

[25] Thomas Zimmermann, Peter Weissgerber, Stephan Diehl, and Andreas Zeller. Mining version histories to guide software changes. *IEEE Trans. Softw. Eng.*, 31(6):429–445, June 2005.

[26] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, November 1975.

[27] Ricardo A. Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[28] Sandeep Tata and Jignesh M. Patel. Estimating the selectivity of tf-idf based cosine similarity predicates. *SIGMOD Rec.*, 36(2):7–12, June 2007.

[29] G. J. F. Jones and P. J. Brown. The role of context in information retrieval. `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.94.2124&rep=rep1&type=pdf`, 2004.

[30] Vishnu Challam, Susan Gauch, and Aravind Chandramouli. Contextual search using ontology-based user profiles. In *Large Scale Semantic Access to Content (Text, Image, Video, and Sound)*, RIAO '07, pages 612–617, Paris, France, France, 2007. LE CENTRE DE HAUTES ETUDES INTERNATIONALES D'INFORMATIQUE DOCUMENTAIRE.

[31] Bruno Antunes, Joel Cordeiro, and Paulo Gomes. Context-based search in software development. In *ECAI*, pages 937–942, 2012.

[32] Carsten Keßler. *Context-aware semantics-based information retrieval*. PhD thesis, University of Münster, 2010. http://d-nb.info/1007472324.

[33] Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, and Mike Dean. Swrl: A semantic web rule language combining owl and ruleml. W3c member submission, World Wide Web Consortium, 2004.

[34] Ahu Sieg, Bamshad Mobasher, and Robin Burke. Ontological user profiles for representing context in web search. In *Proceedings of the 2007 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology - Workshops*, WI-IATW '07, pages 91–94, Washington, DC, USA, 2007. IEEE Computer Society.

[35] Shun Hattori, Taro Tezuka, and Katsumi Tanaka. Activity-based query refinement for context-aware information retrieval. In Shigeo Sugimoto, Jane Hunter, Andreas Rauber, and

Atsuyuki Morishima, editors, *ICADL*, volume 4312 of *Lecture Notes in Computer Science*, pages 474–477. Springer, 2006.

[36] Jiawei Han. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

[37] Roy Thomas Fielding. *REST: Architectural Styles and the Design of Network-based Software Architectures*. Doctoral dissertation, University of California, Irvine, 2000.

[38] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by gibbs sampling. `http://nlp.stanford.edu/~manning/papers/gibbscrf3.pdf`, 2005.

[39] Ana Rita Bento. *Semantic Social Mining*. Master thesis, University of Coimbra, Coimbra, 2014.