Departamento de Engenharia Informática

Relatório de TRABALHO FINAL DE CURSO

LICENCIATURA EM ENGENHARIA INFORMÁTICA E DE COMPUTADORES (LEIC)

Ano Lectivo 2006 / 2007

N.º da Proposta: 32

Título: Sinónimos Só Atrapalham

Professor Orientador:

Pável Calado _____

Co-Orientador:

Andreas Wichert _____

Alunos:

48278, João Miranda _____

50298, Fernando Ribeiro _____

Agradecimentos

Gostaríamos de agradecer ao Professor Pável Calado a orientação e o apoio que nos deu na realização deste trabalho. A sua disponibilidade e acompanhamento foram fundamentais para levar a bom porto os desafios propostos.

Queremos agradecer aos nossos colegas Luís Maranga e Francisco Babo pela proximidade e sugestões prestadas ao longo deste último ano que se revelaram úteis na execução deste trabalho.

Este trabalho é um culminar de um percurso difícil e trabalhoso. Um agradecimento especial aos colegas Pedro Asseiceiro, Paulo Marques, António Ferreiro, Tiago Lucas e Rui Curto pelo companheirismo, atenção e ensinamentos que nos deram ao longo dos anos que nos acompanharam no Instituto Superior Técnico.

Queremos agradecer aos nossos familiares mais próximos e a todos os nossos amigos pela paciência e força que nos infundiram quando mais necessitámos.

Para terminar, gostaríamos de agradecer a todos os que colaboraram com críticas, sugestões e testes, que permitiram tornar melhor o resultado final deste trabalho.

Resumo

Uma busca textual por "Idade Média" apresentará documentos onde surjam as palavras "Idade" e "Média". Documentos sobre a peste negra, as cruzadas ou o estilo gótico não surgirão, em virtude de neles não constarem explicitamente as palavras Idade e Média. Para que tal fosse possível, seria necessário partir para uma abordagem semântica.

O trabalho desenvolvido consistiu na aplicação do modelo vectorial para correlacionar documentos entre si. Desta forma, é possível estabelecer-se uma relação semântica quantificada em afinidades. O sistema desenvolvido foi materializado num motor de busca (Babuska), permitindo aplicar técnicas de procura textual, semântica e combinada a um conjunto de documentos previamente tratado. Como base do motor de busca foi utilizado o *Lucene* [6,7].

Foram constituídas várias colecções para testar a validade dos algoritmos desenvolvidos. As colecções que melhores resultados originaram foram as colecções da *Wikipédia* (portuguesa e inglesa). Constatou-se que os resultados variam consoante o conteúdo da colecção e o idioma da expressão de busca.

Os resultados obtidos foram positivos, mostrando que se consegue pesquisar em função do significado das palavras e não apenas da sua componente textual. Isto significa que é possível obter documentos que estejam relacionados com a expressão de busca sem que ela surja de forma explícita no texto. O trabalho desenvolvido mostrou que é possível incorporar sistemas de pesquisa semântica em motores de busca.

Palavras-chave: pesquisa, semântica, modelo vectorial, documentos, afinidades, motor de busca, Babuska, colecções.

Abstract

Searching for “Middle Ages” in any search engine will return documents containing the words “Middle” and “Ages”. Documents regarding the black plague, the crusades, or the gothic style will not be retrieved, as they do not explicitly exist in the documents. Thus, for this to be possible, it would be necessary to follow a semantic approach.

The present work makes use of the space vector model to obtain similarities between documents. A search engine (Babuska) was developed, having the same features of a regular search engine, but also allowing a semantic and combined search on a set of documents. The core of Babuska is based on Lucene [6,7].

Some collections were made in order to test the developed algorithms. Results for the portuguese and english Wikipedia proved to be the best ones. Results changed with collection’s content and query’s language.

The results were promising, showing that semantic search is feasible. Retrieving documents related to a query without it being explicitly in the document is indeed possible.

Keywords: search, semantic, space vector model, documents, similarity, search engine, Babuska, collections.

Índice

Índice de Figuras

Índice de Tabelas

1. Introdução

As pesquisas que se efectuam nos motores de busca tradicionais como o *Google*, o *Yahoo*, entre outros, não eliminam o problema da comparação textual das palavras. Por exemplo, quem procura, no *Google*, por "carros" certamente também gostaria de ver páginas com "automóveis". Do mesmo modo, quem procura "Casa Branca" certamente não está interessado numa casa branca qualquer. Estes problemas, entre outros que se prendem com o significado e as relações entre palavras, são dos maiores causadores de resultados irrelevantes nos motores de busca. Isto acontece porque, nestas ferramentas, não existe uma noção de semântica associada às palavras, que são comparadas apenas textualmente. De forma a evitar estes problemas, os métodos de pesquisa devem ser modificados de forma a permitir procurar não uma palavra mas também o seu contexto semântico.

É cada vez maior a quantidade de informação disponível ao utilizador num contexto como a *Internet*, exemplo paradigmático da sociedade da informação. As pesquisas textuais não resolvem por si só o problema de uma determinada procura.

Suponhamos que alguém procura por “pentagrama” ou “casa branca”. O que se pretende obter?

Ao pesquisar por informação específica relacionada com determinado assunto o utilizador pretende obter resultados que não se limitem à comparação textual das palavras. Uma palavra pode ter vários significados ou contextos semânticos que a busca textual não consegue discernir. Por exemplo, um músico quando pesquisa por “cravo” não quer flores mas obras musicais para cravo ou resultados relacionados com o instrumento musical.

Quando se procura por um artigo de futebol utilizando a expressão de busca “futebol”, uma busca textual devolverá resultados de documentos em que a palavra “futebol” aparece. Documentos que sejam sobre futebol mas não contenham explicitamente a palavra “futebol” não aparecerão: documentos que falem sobre equipas, jogos, resultados, treinadores, jogadores, árbitros, mas em que “futebol” não conste, não aparecerão. O objectivo de uma busca semântica será perceber que esses documentos estão relacionados com futebol e que, por isso, surjam numa pesquisa semântica por “futebol”, ao contrário do que sucede numa busca textual.

Desenvolveu-se uma abordagem à busca semântica estabelecendo relações entre documentos e classes (a categoria a que um documento pertence, por exemplo, um documento sobre futebol pertence a desporto) para que uma pesquisa seja feita semanticamente pelas palavras que surgem no documento e na expressão de busca e não apenas por comparação textual da expressão de busca em cada documento.

A partir de uma colecção de documentos classificaram-se esses documentos e, posteriormente, usou-se esta classificação para calcular a afinidade entre uma expressão de busca e os documentos classificados.

Os resultados obtidos foram positivos tal como descrito no capítulo 5. As pesquisas semânticas e combinadas permitiram melhorar os resultados em relação às pesquisas textuais. Os algoritmos desenvolvidos mostram uma boa eficácia na obtenção dos resultados.

2. Conceitos, Técnicas e Metodologias

2.1 Modelo Vectorial

A utilização do modelo vectorial [1] permite efectuar recuperação de informação, indexação e seriação de resultados. Um vector é uma forma de representação de documentos contendo palavras indexadas. Estes documentos são usados para procuras nos motores de busca. Os pesos para cada resultado de uma busca podem ser calculados através do ângulo feito entre cada vector de documentos e o vector originado pela busca. Um exemplo da representação destes vectores pode ser observado na Figura 2.1.

A busca é feita usando palavras-chave inseridas pelo utilizador. Estas palavras podem ser vistas como um documento, ou seja, também como um vector. É este vector que é comparado com os restantes vectores existentes na colecção.

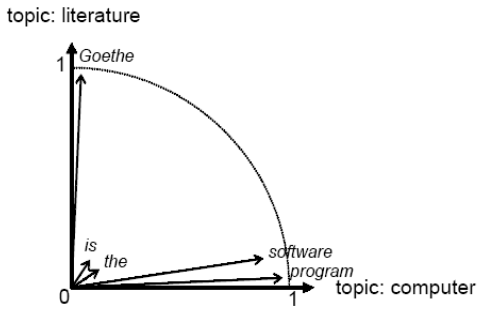

Figura 2.2 - Interpretação dos vectores de palavras no modelo vectorial [3].

O objecto de comparação não é o ângulo em si mas o co-seno do ângulo formado pelos vectores. Se o co-seno do ângulo originado for zero, os documentos não têm similaridade entre si.

O modelo vectorial pode apresentar algumas limitações [9]:

1. Documentos com demasiado texto têm mau desempenho uma vez que podem apresentar valores de afinidade baixos;
2. É necessário garantir que as palavras sejam pesquisadas como palavras inteiras e não como parte de outras palavras. Caso contrário, podem ser originados falsos positivos. Por

exemplo, “desempenho” não deverá aparecer identificada nos resultados da pesquisa por “empenho”;

3. Documentos com contexto similar mas com conteúdo textual diferente não são identificados como relacionados, podendo originar um falso negativo. Por exemplo, documentos sobre automóveis onde nunca apareça a palavra “carro” e documentos sobre carros onde nunca apareça a palavra “automóvel” poderão não ter afinidade.

2.2 Aplicação do Modelo Vectorial ao processo de pesquisa semântica

Para se aplicar o modelo vectorial a este trabalho definiu-se um conjunto de treino e teste, onde o conjunto de treino servirá para determinar os documentos das classes e o conjunto de teste servirá para se obter as afinidades entre o conjunto de documentos de treino (mais concretamente dos documentos das classes) e o conjunto de documentos de teste. Isto será explicado mais detalhadamente na secção 4.1.1.

O procedimento para representar um documento é o seguinte:

- Para cada documento conta-se o número de ocorrências de cada palavra;

- Contam-se todos os documentos do conjunto de treino;

- Aplica-se, para cada palavra, a seguinte fórmula para se calcular o seu peso (w) no documento:

$$w = freq_k \times \ln \frac{N+1}{N_k} \qquad (1),$$

onde $freq_k$. é o número de ocorrências da palavra k no documento i, N é o número total de documentos, N_k é o número de documentos onde a palavra k existe.

Neste trabalho, as classes de documentos são representadas também por vectores. Estes vectores não são mais que as médias dos vectores dos documentos que pertencem à classe.

Para representar os documentos tendo em conta o contexto semântico das palavras, estes são representados como vectores de classes em vez de vectores de termos. Para tal, a similaridade do vector original do documento com cada vector de classes é calculada e é usada como uma coordenada desse vector. Por exemplo, no seguinte vector, que corresponde a um documento

pertencente à classe computador, há uma afinidade de 0,4 com a classe computador, 0,4 de afinidade com a classe *software*, 0,1 com a classe *hardware* e assim sucessivamente:

computador computador 0,4 software 0,4 hardware 0,1 internet 0,05 home 0,01

A afinidade entre os documentos e as classes é obtida da seguinte forma:

$$\sum \frac{k_{treino} \times k_{teste}}{\|N_{treino}\| \times \|N_{teste}\|} \qquad (2),$$

onde, para cada palavra *k* de cada documento do conjunto de teste, se procura a palavra *k* no conjunto de treino. É extraído o peso dessas palavras. Multiplicam-se estes valores e soma-se o valor obtido para as restantes palavras comuns no conjunto de treino e conjunto de teste. Seguidamente divide-se pelas normas dos vectores correspondentes. O procedimento anterior é repetido para todos os documentos do conjunto de teste.

2.3 Indexação

Um índice é uma lista de palavras e ponteiros que permitem associar palavras a documentos [15]. Há dois tipos principais de índices: os índices directos (*forward index*) e os índices invertidos (*inverted index*).

Num índice directo listam-se os documentos e as palavras que pertencem a cada documento (Tabela 2.1).

Tabela 2.1 – Exemplo de um índice directo.

Documento	Palavras
1	água, carro, óleo
2	água, torneira, copo
3	óleo, torneira, carro

Num índice invertido listam-se as palavras e os documentos em que elas aparecem (Tabela 2.2).

Tabela 2.2 – Exemplo de um índice invertido.

Palavra	Documentos
água	1, 2
carro	1, 3
óleo	1, 3
torneira	2, 3
copo	2

Os índices invertidos são muito usados no modelo vectorial.

Uma busca textual resulta da procura dos termos existentes na expressão de busca nos documentos indexados.

2.4 Tecnologias e ferramentas

As tecnologias e ferramentas utilizadas no trabalho desenvolvido encontram-se descritas de seguida.

2.4.1 *JSP*

As *JSP* (*Java Server Pages*) são uma das tecnologias utilizadas para a realização deste trabalho, nomeadamente no motor de busca Babuska (motor de busca desenvolvido no âmbito deste trabalho e que será descrito pormenorizadamente na secção 4.3). Esta tecnologia permite a utilização de código *JAVA*, necessário para a realização do trabalho, uma vez que o motor de busca é baseado numa tecnologia escrita em *JAVA*, o *Lucene* (ver 2.4.3).

A estrutura do Babuska é baseada em páginas escritas em *HTML*, com código *JAVA* e *JAVA Script*. Deste modo, foi possível utilizar-se o código escrito em *JAVA* para manipulação de resultados e apresentar em formato *HTML* esses resultados nas *JSP*.

Para que as páginas possam ser utilizadas é necessário compilar e interpretar as páginas utilizando para o efeito o *Apache Tomcat* (ver 2.4.2), onde o motor de busca poderá ser acedido. O *Tomcat* procede à compilação do código definido nas *JSP*.

2.4.2 *Apache Tomcat*

O *Apache Tomcat* [5] é um sistema que integra as especificações das *JSP* definidas pela *Sun Microsystems* e que permitem correr o código definido nas *JSP* do motor de busca Babuska. A utilização deste sistema permite o correcto funcionamento do Babuska e da sua interacção com o utilizador.

2.4.3 *Lucene*

Para o Babuska funcionar, para além do uso de *JAVA*, *JSP*, *Tomcat*, é necessário um indexador de documentos que permita pesquisar e obter resultados para qualquer pesquisa efectuada nesses documentos. O *Lucene* é a base do Babuska, pois o *Lucene* permite fazer pesquisas a documentos indexados, devolvendo como resultado um conjunto de documentos que satisfaçam a pesquisa. Uma vez que o *Lucene* faz pesquisas textuais como qualquer outro motor de busca, não permite obter resultados semânticos directamente. Estes são conseguidos transformando a expressão de busca numa expressão de busca semântica, como descrito no capítulo 4.4.

O *Lucene* é um motor de busca, escrito em *JAVA*, com funcionalidades de alto nível de indexação e de pesquisa textual. É um projecto *open-source* e disponível para utilização livre [6].

O funcionamento do *Lucene* baseia-se na definição de documento e de campos (*fields*). O documento contém vários campos que compõem os diferentes elementos da indexação efectuada a um documento. Por exemplo, um documento do *Lucene* pode conter, entre outros, o campo do nome e do resumo. É possível adicionar-se campos ao *Lucene* de acordo com as necessidades. Para mais informações acerca do *Lucene* consultar o endereço [6].

A indexação que se obtém a partir do *Lucene* é baseada em indexadores (já existentes ou criados para o efeito) como é o caso do *HTMLIndexer* que permite indexar ficheiros *HTML, o NormalIndexer* que permite indexar as palavras de um documento excluindo as *stop words*, entre outros. Cada um destes indexa cada documento de acordo com as suas especificações. A título de exemplo, o indexador utilizado no nosso trabalho é o *NormalIndexer*.

O *Lucene* permite fazer pesquisas de forma semelhante aos outros motores de busca, possibilitando a utilização de conjunções (OR), disjunções (AND), entre outros [7-8].

O *Lucene* incorpora o conceito de *boost*, isto é, permite dar um valor mais elevado a uma palavra quando se faz uma pesquisa, ou seja, dar maior importância a essa palavra. Por exemplo, dar um *boost* de 3 a uma palavra corresponde a dar uma importância três vezes superior ao que ela teria normalmente. No caso da pesquisa semântica, esta possibilidade foi utilizada, como é explicado na secção 4.4.3.

2.4.4 *Httrack*

O *Httrack* [11] é um programa que permite descarregar um *site* reconstruindo localmente todo o conteúdo disponível *on-line*. Com diversas opções disponíveis e amplamente configurável, permite efectuar cópias locais de recursos da *Internet* através do varrimento de todas as ligações disponíveis no recurso. Os recursos podem ser depois navegados localmente. As opções permitem escolher, por exemplo, os tipos de recursos a descarregar, como imagens, arquivos, documentos de texto, entre outros, e a profundidade pretendida, ou seja, quantos níveis internos ou externos podem ser seguidos a partir do endereço original.

O *Httrack* permite definir que extensões devem ser deixadas de fora ou que ficheiros devem ser incluídos. Possibilita a definição de diversos parâmetros como os limites de velocidade de transferência, o limite máximo de ficheiros a descarregar ou a inclusão de ficheiros externos ao local de origem, como, por exemplo, ficheiros comprimidos.

O *Httrack* foi utilizado neste trabalho por possibilitar, de forma relativamente fácil, a obtenção dos dados necessários para constituir as colecções destinadas a serem utilizadas. Desta forma, descarregou-se parte dos recursos do *Dmoz* e da *Wikipédia* inglesa e portuguesa para construção de colecções para utilização no sistema.

3. Metodologia de trabalho

A resolução de um trabalho complexo, próprio da natureza de um trabalho final de curso, obriga a um planeamento e gestão de tempo eficazes, de modo a atingir os resultados esperados.

O plano do trabalho incluiu as seguintes tarefas:

- Ler bibliografia sobre modelo vectorial e indexação de documentos (09/2007)
- Obter e explorar as colecções iniciais (*Reuters* e *Cadê*) (10/2007)
- Desenvolver os algoritmos de cálculo de afinidades (11/2007)
- Fazer algumas consultas de teste (01/2007)
    - Criar consultas (olhando para o conteúdo da colecção)
    - Para cada consulta:
        - Transformar as consultas em vectores de afinidades (similaridades com as médias)
        - Medir a afinidade de cada documento com a consulta
        - Ordenar os documentos por afinidade
        - Mostrar os N primeiros
        - Ver se os resultados são aceitáveis
- Desenvolver um sistema de busca (06/2007)
    - Estudar o *Lucene* para ver se pode ser usado
        - O *Lucene* pode sempre ser usado para consultas normais
    - Se não puder, desenvolver o sistema:
        - Criar o "transformador" de documentos (02/2007)
            - Dado um conjunto de treino (documentos e as suas classes):
                - calcular os vectores dos documentos
                - calcular os vectores das classes (médias)
                - calcular os novos vectores do documentos (afinidades com as médias)
        - Desenvolver um indexador de documentos (03/2007)
            - Dados os vectores dos documentos o indexador deve criar um ficheiro invertido

        - Desenvolver um componente de *ranking* (04/2007)
            - Dado um conjunto de palavras-chave (consulta)
                - Transformar num vector de afinidades
                - Usar o ficheiro invertido para gerar um *ranking*
        - Desenvolver um componente de avaliação (05/2007)
            - Dados uma consulta, um conjunto de resultados e um conjunto de documentos relevantes, calcular precisão.
        - Desenvolver uma interface para o sistema (06/2007)
- Recolher e explorar colecções novas (*Dmoz*, *Wikipédia* portuguesa e inglesa) (07/2007)
- Realizar um conjunto de testes e avaliar os resultados obtidos (08/2007)
- Redigir o relatório (09/2007)

A actividade foi cumprida de acordo com o plano estabelecido inicialmente entre o orientador e os elementos do grupo do TFC (Trabalho Final de Curso).

A criação de um motor de busca foi feita com recurso à utilização do *Lucene*, que se mostrou adequado às necessidades do projecto. Caso não tivesse sido possível utilizar o *Lucene*, ter-se-ia partido para a criação de um motor de busca próprio, conforme enunciado no planeamento descrito.

4. Descrição do trabalho

Ao fazer procuras num motor de busca obtêm-se tradicionalmente resultados textualmente relacionados com a palavra introduzida. No entanto, se pretendermos obter resultados relacionados com a semântica das palavras desejadas, é necessário efectuar algumas alterações à abordagem normalmente utilizada pelos motores de busca tradicionais.

A realização do projecto passa por compreender como as palavras se relacionam. Para que tal seja possível, um conjunto de documentos, em que cada documento pertence a uma classe (categoria) e cada classe tem um conjunto de palavras, é processado de forma a obter a sua relação com os documentos da mesma classe e com as palavras que compõem essas classes. Este procedimento indica quais as afinidades que cada documento e cada palavra tem com cada uma das classes.

Para fazer buscas, as palavras-chave inseridas pelo utilizador podem ser vistas novamente como um documento.

4.1 Descrição dos ficheiros relevantes para o trabalho

O funcionamento correcto do sistema baseia-se na utilização de um conjunto de diferentes ficheiros. Cada um destes ficheiros tem uma especificidade que contribui para o correcto funcionamento do sistema. De seguida descreve-se qual a função de cada um destes ficheiros.

4.1.1 Documentos de treino e teste

Para que seja possível fazer-se pesquisas semânticas aplicando o modelo vectorial descrito em 2.2 é necessário treinar o sistema para que este consiga funcionar. Deste modo, é necessário criar-se um conjunto de documentos que vão ser utilizados no seu treino. Este conjunto tem de ser representativo e o mais completo possível para um bom desempenho do sistema.

O conjunto de teste é necessário para o teste e validação do sistema. Este conjunto de teste também tem de ser bastante representativo no sistema, porque as pesquisas serão efectuadas sobre o conjunto de teste. Se o teste for pequeno os resultados obtidos serão limitados.

Os conjuntos de treino e teste foram obtidos separando os documentos de cada colecção utilizada (ver descrição das colecções na secção 4.2). Esta separação foi efectuada atribuindo um valor de 30 % para o conjunto de treino e 70 % para o conjunto de teste.

A divisão foi efectuada por documentos dentro de cada classe, ou seja, uma classe com 100 documentos daria origem a 30 documentos de treino e 70 de teste. Entendeu-se que a separação de 30 % e 70 % seria uma boa separação pois permitia um processamento mais leve no consumo de recursos (memória e processador). Estes parâmetros foram mantidos durante todo o trabalho.

4.1.2 Ficheiros de classes, afinidades e endereços

Para que seja possível a pesquisa semântica é necessário que algumas modificações sejam efectuadas nos ficheiros de treino e teste descritos em 4.1.1.

O ficheiro de classes corresponde à transformação do ficheiro contendo os documentos de treino tal como descrito no capítulo 2.2. Este ficheiro de classes irá conter todas as classes individualizadas (uma por linha) com as respectivas palavras que compõem essa classe mais o peso para cada uma delas.

O ficheiro de afinidades corresponde à transformação do ficheiro contendo os documentos de teste num ficheiro de afinidades. A transformação é efectuada tal como descrito no capítulo 2.2. Este ficheiro irá conter todos os documentos de teste e as classes com as quais têm afinidade, juntamente com o respectivo valor de afinidade. Este ficheiro contém os documentos onde a pesquisa semântica é efectuada.

As colecções do *Dmoz* e da *Wikipédia* (tal como descrito em 4.2) são obtidas a partir de páginas da *Internet*. Para que seja possível referenciar os documentos originais é necessário a existência de um ficheiro (ficheiro de endereços) que permita fazer a correspondência entre os documentos encontrados numa pesquisa e o respectivo endereço da página. Cada linha deste ficheiro contém um número, um endereço e um título que corresponde a um documento presente no ficheiro de afinidades e no ficheiro dos documentos de teste.

4.2 Colecções

Na elaboração do trabalho foi necessário utilizar colecções de modo a testar o sistema e afinar os algoritmos desenvolvidos. Estas colecções foram criadas a partir de diversas fontes, a *Reuters* e o *Cadê*, obtidas de [4], o *Dmoz* obtida de [13] e a *Wikipédia* (portuguesa e inglesa) [12].

Cada colecção utilizada contém um número de documentos e classes diferente, de acordo com a Tabela 4.1.

Tabela 4.1 – Colecções utilizadas no desenvolvimento do trabalho.

Colecção		Número de Classes	Número de Documentos
Reuters		8	7674
Cadê	Menor	188	36911
	Maior	1030	41675
Dmoz		478	11113
Wikipédia Portuguesa		7379	38213
Wikipédia Inglesa		17733	35286

As colecções da *Reuters* e do *Cadê* não sofreram alterações aos ficheiros obtidos para processamento. Pelo contrário, as colecções do *Dmoz* e da *Wikipédia* passaram por um pré-processamento antes de poderem ser utilizadas.

4.2.1 Processamento das Colecções da *Reuters* e do *Cadê*

O processamento destas colecções cingiu-se à separação dos documentos em treino e teste e posteriormente à construção dos ficheiros de classes e afinidades, como descrito no Anexo H.

4.2.2 Processamento das Colecções da *Wikipédia* e *Dmoz*

O processamento destas colecções envolve um maior número de passos devido à maior complexidade dos documentos que são obtidos das páginas da *Wikipédia*. Depois de se obterem os documentos com a ajuda do *Httrack* (ver secção 2.4.4) extrai-se o conteúdo útil de cada documento. Criam-se em seguida os ficheiros de classes, afinidades e endereços, tal como descrito no Anexo H.

4.3 Babuska

Um dos objectivos do trabalho é aplicar todo o sistema desenvolvido a um motor de busca de forma a poder obter resultados práticos do trabalho efectuado. Para o efeito, foi utilizado como base o *Lucene*, um sistema de pesquisa, que permite a indexação e pesquisa em ficheiros. Foram desenvolvidas classes em *Java* que permitissem adaptar as capacidades do *Lucene* às necessidades do trabalho. Foi desenvolvida uma interface *web* que permite avaliar as potencialidades do sistema, efectuando pesquisas textuais, semânticas e combinadas.

O Babuska é um motor de busca desenvolvido com base no motor do *Lucene*, que permite efectuar buscas nas colecções utilizadas (*Dmoz*, *Wikipédia* portuguesa e *Wikipédia* inglesa).

O sistema tem três tipos de pesquisa. A pesquisa textual é uma pesquisa simples, onde o sistema procura por documentos que contenham explicitamente as palavras constantes na expressão de busca. A pesquisa semântica efectua uma pesquisa para além da pesquisa textual, através dos mecanismos descritos na secção 4.4.3. A pesquisa combinada permite combinar as pesquisas textual e semântica, dispondo os resultados conforme a pontuação que tiver sido obtida fruto das opções escolhidas para a pesquisa.

Na Figura 4.1 pode observar-se o aspecto geral do Babuska. Este é composto por 3 áreas. A área de pesquisa, onde se introduz uma expressão de busca, a área da escolha do tipo de busca e a área das preferências.

Figura 4.1 – Motor de busca Babuska.

4.3.1 Preferências

Para controlar as opções dos tipos de busca e os resultados obtidos poder-se-á recorrer às opções do sistema. Para as pesquisas textuais o número de resultados a apresentar simultaneamente é o único ponto a controlar. Para os restantes tipos de pesquisa poderá ser necessário ajustar outros parâmetros de acordo com as preferências dos utilizadores.

Como mostrado na Figura 4.2, existem três campos para as preferências:

1. **Número de Resultados** – Este campo apresenta a possibilidade de especificar quantos resultados podem aparecer por página e, em caso de estar em modo de avaliação (modo que permite ao utilizador avaliar os resultados obtidos nas pesquisas efectuadas), quantos resultados queremos avaliar no máximo. Os valores por omissão são 10 para cada um dos campos.
2. **Pesquisa Semântica** – Este campo permite definir qual o valor a utilizar na multiplicação do valor da afinidade, ou seja, corresponde ao factor multiplicativo que está descrito na secção 4.4.3.
3. **Pesquisa Combinada** – Este campo permite ao utilizador definir as opções para a busca combinada. A pesquisa combinada pode ser efectuada utilizando uma soma dos valores, a média dos valores ou uma multiplicação dos valores das pesquisas textuais e semânticas. Os valores dos campos *Peso* permitem definir a importância da pesquisa textual e semântica na pesquisa combinada.

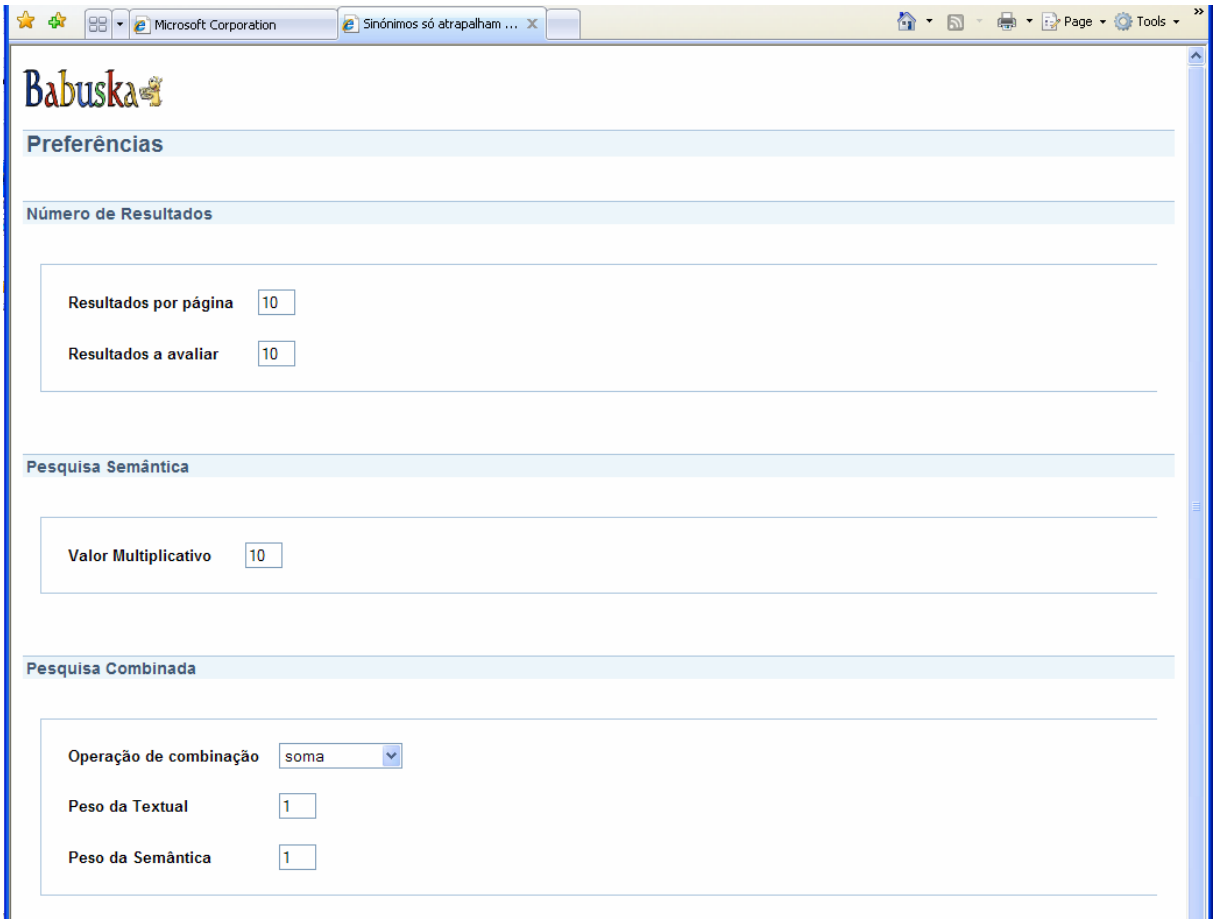

Figura 4.2 – Preferências do motor de busca Babuska.

4.3.2 Pesquisa Textual, Semântica e Combinada

Todas as pesquisas obtidas no Babuska apresentam o mesmo aspecto. Por exemplo, uma pesquisa textual à expressão "amor de perdição" dá origem ao ecrã representado na Figura 4.3.

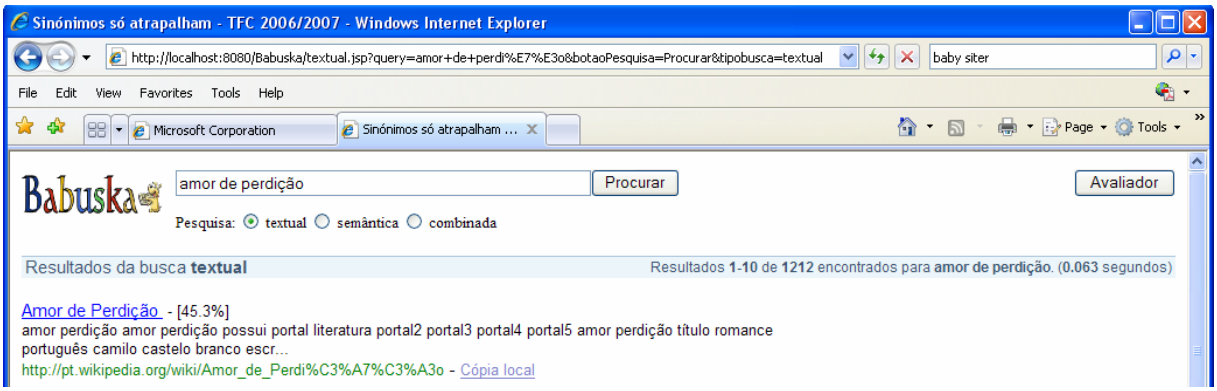

Figura 4.3 – Exemplo de uma pesquisa no Babuska.

Cada resultado surge com o título, pontuação, resumo, endereço e uma cópia local do ficheiro em disco. A parte inferior do ecrã permite navegar entre as diferentes páginas de resultados, e possibilita a introdução de uma outra expressão, como se pode verificar na Figura 4.4.

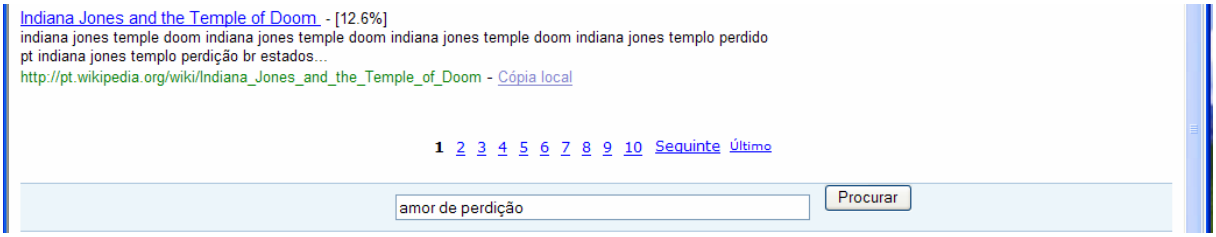

Figura 4.4 – Parte inferior da página de resultados do Babuska.

A busca textual é uma busca tradicional semelhante a um vulgar motor de busca.

A busca semântica é um pouco mais complexa, uma vez que é necessário transformar a expressão de busca numa expressão semântica. O modo como a transformação é feita baseia-se na classificação da expressão como se fosse um documento e na obtenção das afinidades que a expressão tem com as classes existentes.

Devido à complexidade das pesquisas semântica e combinada, o tempo de processamento pode ser longo.

A pesquisa combinada é uma combinação das duas pesquisas anteriores, ou seja, os resultados são dados em função dos resultados da textual e da semântica. Apenas se consideram as primeiras 500 ocorrências para cada uma das pesquisas. Desta forma evita-se um tempo de processamento demasiado elevado.

Para exemplificar o funcionamento da pesquisa combinada, considere-se o seguinte exemplo, em que A, B, C e D são os documentos devolvidos pelo sistema:

- Pesquisa textual com A [70%], B [50%], C [2%], D [2%],
- Pesquisa semântica com C [80%], A [70%], D [60%], B [2%]
- A combinada irá corresponder a (para a média): A [0,7], C [0,41], D [0,31], B [0,26]

Como se pode verificar, os resultados podem ser alterados de forma significativa, dependendo dos valores que obtiveram nas diferentes pesquisas.

4.3.3 Modo de Avaliação de resultados

A análise dos resultados obtidos só será válida caso haja uma avaliação qualitativa e quantitativa dos mesmos. O modo de avaliador fica activo quando se prime o botão **Avaliador** existente no canto superior direito da interface do motor de busca, tal como se pode verificar na Figura 4.5. Esta figura mostra como a página de resultados do Babuska fica quando se prime o botão **Avaliador**. Neste caso, o nome do botão passa para **Utilizador**, no caso do utilizador querer voltar ao modo de navegação normal. Surgem então novas opções que permitem avaliar, registar o resultado da avaliação (**Registar**) e ver os resultados de todas as avaliações anteriores (**Resultados**). Para se seleccionar um resultado como bom deverá marcar-se a caixa adjacente ao resultado. Os resultados obtidos são escritos num ficheiro cujo conteúdo é semelhante ao da Figura 4.6.

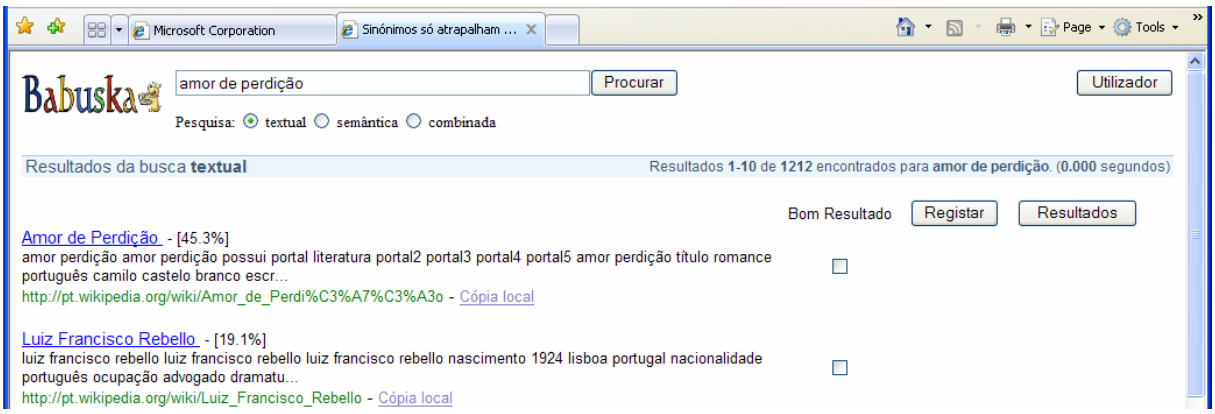

Figura 4.5 – Modo de Avaliação de resultados das buscas.

```
#AVALIAÇÃO=================================================

QUERY:  fast food
TIPO:   textual
RESULTADOS BONS:        5
RESULTADOS AVALIAVEIS:  10
RESULTADOS ENCONTRADOS: 178
DATA:   08-09-2007 02:25:16
IP:     127.0.0.1

#DADOS DOS RESULTADOS

0 - Fast-food  - [100 %]
http://pt.wikipedia.org/wiki/Refei%C3%A7%C3%A3o_r%C3%A1pida
C:\Indexado\Normal\23248.txt

1 - Salgado (comida)  - [60.4 %]
http://pt.wikipedia.org/wiki/Salgado_(comida)
C:\Indexado\Normal\26316.txt

4 - McDonald's  - [51.3 %]
http://pt.wikipedia.org/wiki/McDonald%27s
C:\Indexado\Normal\9825.txt

6 - Nutrição  - [47.9 %]
http://pt.wikipedia.org/wiki/Nutri%C3%A7%C3%A3o
C:\Indexado\Normal\5693.txt

9 - Gastronomia  - [46.2 %]
http://pt.wikipedia.org/wiki/Gastronomia
C:\Indexado\Normal\11359.txt

#FIM=======================================================
```

Figura 4.6 – Ficheiro contendo os resultados das avaliações efectuadas.

4.4 Descrição do processo de transformação de documentos e expressões de busca em afinidades

De seguida descreve-se o processo de transformação de documentos e expressões de busca em afinidades, incluindo a criação dos ficheiros dos documentos contendo as classes, a separação dos ficheiros de teste e de afinidades, e a obtenção das afinidades entre uma expressão de busca e as classes definidas no ficheiro de classes.

4.4.1 Criação dos ficheiros dos documentos contendo as classes, e dos documentos de afinidades

A criação do ficheiro contendo os documentos de classes é feita como descrito em 2.2, aplicando a equação (1). Todo o processo é feito utilizando o programa descrito no Anexo A.

Considere-se o seguinte exemplo, obtido de parte da colecção da *Reuters* (Figura 4.7):

```
earn   champion products approves stock split champion products inc board …
acq    computer terminal systems cpml completes sale computer terminal systems …
acq    hong kong firm ups wrather wco stake pct industrial equity pacific hong kong …
earn   cobanco inc cbco year net shr cts dlrs net assets mln mln deposits mln mln …
earn   international inc qtr jan oper shr loss two cts profit cts oper shr profit …
earn   brown forman inc bfd qtr net shr dlr cts net mln mln revs mln mln mths shr …
earn   dean foods sees strong qtr earnings dean foods expects earnings for fourth …
acq    chemlawn chem rises hopes for higher bids chemlawn corp chem attract …
trade  brazil anti inflation plan limps anniversary inflation plan initially hailed …
```

Figura 4.7 – Alguns documentos do conjunto de treino.

Cada linha anterior representa um documento, em que a primeira palavra corresponde à classe a que o documento pertence e as restantes correspondem às palavras que compõem o documento. Na criação das classes agregar-se-ão as classes com o mesmo nome, bem como as palavras pertencentes a essas classes. No final do processamento obtém-se um conjunto de documentos que terá uma classe por linha, tal como se mostra na Figura 4.8:

```
1 acq    hong 1.54 systems 1.54 chemlawn 1.54 chem 1.54 kong 1.54 computer 1.54 …
2 trade  inflation 4.61 plan 4.61 anti 2.3 initially 2.3 limps 2.3 anniversary 2.3 …
3 earn   mln 2.58 net 1.29 shr 1.2 cts 0.96 earnings 0.92 oper 0.92 products 0.92 …
```

Figura 4.8 – Classes obtidas após transformação do conjunto de treino.

O valor obtido para cada palavra corresponde ao peso dessa palavra na classe respectiva. Cada linha é, portanto, o vector que representa a classe.

Para a obtenção do ficheiro de afinidades é necessário o ficheiro de classes obtido como acima descrito. O ficheiro dos documentos de teste é processado de forma semelhante ao ficheiro dos documentos de treino excepto no que à junção dos documentos da mesma classe diz respeito. O procedimento segue a fórmula (2) descrita na secção 2.2.

Considere-se o exemplo seguinte com quatro documentos do ficheiro de teste (Figura 4.9):

```
earn  george weston year net shr dlrs dlrs net mln mln revs billion
acq   circuit systems csyi buys board maker circuit systems inc bought
earn  amatil proposes two for bonus share issue amatil amaa proposes
earn  bowater pretax profits rise mln stg shr div making turnover
```

Figura 4.9 – Alguns documentos do conjunto de teste.

Estes documentos são processados juntamente com o ficheiro dos documentos de classes acima descrito através do programa em Anexo A para se calcular as afinidades dos documentos com cada classe. Obtém-se um ficheiro de documentos de afinidades, tal como se mostra na Figura 4.10, em que a primeira palavra corresponde à classe, as palavras seguintes correspondem às classes com a qual tem afinidade e os valores correspondem às respectivas afinidades.

```
1 earn  earn 0.42 acq 0.02
2 acq   acq 0.2
3 earn  earn 0.02 trade 0.02
4 earn  earn 0.19 trade 0.1
```

Figura 4.10 – Documentos de teste da Figura 4.9 transformados em afinidades.

Os documentos do teste foram assim transformados em documentos de afinidades.

4.4.2 Separação dos ficheiros de teste e de afinidades

O *Lucene* indexa documentos em ficheiros individuais e não documentos que estejam contidos num mesmo ficheiro (um por linha) como é o caso dos ficheiros de teste e de afinidades criados anteriormente (secção 4.4.1). Por este motivo é necessário separar cada documento em ficheiros individuais através do programa descrito no Anexo E.

Os documentos dos ficheiros de teste são separados directamente, sem alterações, para ficheiros individuais. Os ficheiros de afinidades, por outro lado, necessitam de uma transformação ao seu conteúdo pois este contém um valor que não serve para indexar no *Lucene* e que corresponde ao valor da afinidade. Para contornar este problema, utiliza-se um factor multiplicativo que transforma a afinidade num valor inteiro que irá corresponder ao número de vezes que a palavra (classe) irá ser repetida no ficheiro separado.

Exemplificando, se um documento for descrito como

1 acq acq 0,35 earn 0,21 trade 0,02,

e se se usar um factor multiplicativo de 10 então o documento será definido como

1 acq acq 3 earn 2,

arredondando-se os números à unidade. O ficheiro deste documento irá conter assim três vezes a palavra (classe) acq e duas a earn, ou seja,

acq acq acq earn earn.

4.4.3 Obtenção das afinidades entre uma expressão de busca e as classes definidas no ficheiro de classes

O algoritmo de pesquisa do Babuska, no que à pesquisa semântica diz respeito, passou por uma evolução ao longo do desenvolvimento do trabalho. Este algoritmo permite transformar a expressão de busca textual numa expressão semântica. O *Lucene* é um motor de busca textual, que usa indexação inversa, mas não consegue distinguir o que é uma busca textual de uma busca semântica. Desenvolveu-se um algoritmo de transformação que é utilizado no Babuska e que foi incorporado numa biblioteca com o nome de *processarQuery.jar*. Este algoritmo passou por uma série de melhoramentos e revisões ao longo do trabalho para tornar mais eficiente todo o processo de transformação.

O algoritmo de transformação da expressão de busca textual na expressão semântica é muito semelhante ao algoritmo descrito em 2.2, com a diferença de se utilizar um factor multiplicativo para expandir as classes da expressão semântica, de forma a estabelecer-se a expressão semântica livre de um valor numérico, tal como visto no final da secção 4.4.2.

A primeira versão do algoritmo da transformação da expressão de busca textual não estava optimizada porque qualquer classe que contivesse uma das palavras da expressão de busca textual era incluída na expressão de busca semântica. Foi necessário repensar a forma de fazer esta transformação. Foi assim desenvolvido um novo método, descrito de seguida, que consiste em considerar todas as palavras para o cálculo das afinidades. Se tal não for possível, é calculado o máximo de palavras que podem ser consideradas, obtendo-se as classes que contenham uma qualquer combinação dessas palavras.

Por cada classe do ficheiro de classes existente determina-se qual a afinidade da expressão de busca com a classe obtendo-se uma expressão que é função, não das palavras, mas das classes. Contudo, existem alguns aspectos importantes que importa referir:

- Inicialmente percorrem-se todas as classes para ver se há alguma classe que contenha todas as palavras da expressão de busca.
- Em caso afirmativo, obtêm-se os valores das afinidades das classes como descrito no algoritmo inicial. Os valores inferiores a 0,09 não serão contabilizados por serem demasiado baixos e para evitar obter classes pouco relevantes para a pesquisa.

- Em caso negativo, isto é, se não existir uma correspondência completa, são seleccionadas as classes com maior correspondência de termos. Por exemplo, se numa busca se introduzir as palavras "casa", "campo" e "vivenda", e apenas se encontrar documentos que contêm em simultâneo duas delas, então as classes que se irão processar contêm qualquer combinação de duas das três palavras possíveis. As combinações serão efectuadas no conjunto {(casa, campo), (casa, vivenda), (campo, vivenda)}.

Após transformação da expressão de busca numa expressão de afinidades, é feita uma comparação dos valores obtidos para cada um dos valores de afinidades de onde se faz a seguinte classificação:

1. Valores de afinidade inferiores a 0,09 são descartados, isto é, todas as classes com estes valores de afinidade não entram para a expressão de busca.
2. Para valores acima de 0,1 é atribuída uma pontuação para as classes. Obtendo-se um valor de 0,9 é somado 9 como pontuação, 0,8 adiciona-se 8 e assim sucessivamente até 0,1 com a adição de 1, dividindo-se estes valores por 10.
3. Finalmente é atribuído um *boost* às classes de acordo com os valores obtidos no ponto anterior mais o valor do factor multiplicativo arredondado às unidades. Este *boost* indica ao *Lucene* que a palavra (classe) terá uma importância maior na pesquisa.

Por exemplo, uma expressão de busca "palavraA palavraB" que origine, após transformação, uma expressão com as afinidades "classeA 0,5 classeD 0,3 classeF 0,02" será internamente representada por "classeA 0,55 classeD 0,33". Após aplicação, por exemplo, de um factor multiplicativo de 10 é novamente recalculada a expressão que irá ser "classeA 6 classeD 3". A expressão de busca que o *Lucene* interpretará será "classeA^6 classeD^3". Estes valores 6 e 3 correspondem ao *boost* aplicado.

5. Resultados

5.1 Como se efectuaram as medições do sistema

A avaliação do motor de busca (Babuska) e dos algoritmos desenvolvidos foi efectuada segundo as características a seguir descritas.

Todas as expressões de busca para qualquer das colecções avaliadas foram definidas pelos elementos do grupo. Estas expressões foram consideradas representativas de cada uma destas colecções, como é o caso de "bola de berlim" para a colecção da *Wikipédia* portuguesa e "white house" para a *Wikipédia* inglesa.

O motor de busca foi instalado em servidores em diferentes computadores.

Todos os testes foram realizados atribuindo 128 *megabytes* de *RAM* para a *Java Virtual Machine* na configuração do *Apache Tomcat*.

As colecções utilizadas para demonstrar o modelo vectorial e a possibilidade de se efectuarem pesquisas semânticas foram as mais variadas, desde colecções de documentos da *Reuters*, do *Cadê* e, mais tarde, do *Dmoz* e da *Wikipédia*. As colecções da *Reuters* e do *Cadê* foram obtidas de [4].

A medida utilizada foi a precisão que corresponde à razão entre os resultados avaliados como bons e os resultados devolvidos. Foram utilizados para avaliação os primeiros dez resultados. Por questões de tempo e de recursos, as avaliações apenas foram executadas pelos elementos do grupo.

Para as colecções da *Reuters* e do *Cadê* não se fez, ao contrário da *Wikipédia*, a avaliação quantitativa dos resultados, mas sim qualitativa. Estas colecções não permitem concluir muito acerca da validade dos resultados pois, não se tratando de páginas *html* (que contêm vulgarmente um título), os documentos são identificados com um número, não oferecendo grande informação sobre o seu conteúdo.

5.2 Resultados do *Cadê* e da *Reuters*

A *Reuters* é uma colecção que apenas contém 8 classes diferentes. Esta limitação foi o principal entrave à obtenção de bons resultados porque quaisquer resultados tinham afinidade com todas as classes.

As classes que compõem esta colecção são:

earn, *acq*, *trade*, *ship*, *grain*, *crude*, *interest* e *money-fx*.

Como se pode verificar, a colecção apenas trata de um assunto muito específico (trocas comerciais) o que limita a pesquisa. No entanto, verificou-se que as classes de teste utilizadas para produzir afinidades deram resultados promissores para a aplicação desta técnica. Como os documentos de teste contêm uma pré-categorização, conseguiu-se perceber até que ponto estes documentos se relacionavam com os documentos de classes produzidos a partir dos documentos de treino.

Tabela 5.1 – Afinidades obtidas para os cinco primeiros documentos do conjunto de treino.

Documento	Classe	Afinidades
1	trade	trade 0.59 money-fx 0.21 ship 0.16 acq 0.16 interest 0.15 crude 0.15 grain 0.13
2	grain	grain 0.25 crude 0.08 ship 0.07 trade 0.07 acq 0.07 interest 0.06 money-fx 0.05
3	ship	ship 0.31 trade 0.07 money-fx 0.06 acq 0.06 crude 0.05 interest 0.05
4	acq	acq 0.15 interest 0.13 money-fx 0.13 trade 0.09 earn 0.07 crude 0.06 ship 0.06
5	earn	acq 0.13 earn 0.09 interest 0.08 money-fx 0.06 trade 0.05 ship 0.04 crude 0.04

Uma vez que os documentos da *Reuters* continham poucas classes, seria necessário utilizar colecções mais representativas e variadas de modo a poder aferir mais correctamente os resultados obtidos. As colecções que se seguiram foram as do *Cadê*, em duas versões, uma com 188 classes e outra com 1030. As diferenças são significativas entre as colecções da *Reuters* e do *Cadê* porque quanto mais classes existirem menos possibilidade existe de os resultados se concentrarem num número reduzido de classes.

As colecções do *Cadê* utilizadas são semelhantes à colecção da *Reuters*.

A análise que a seguir se apresenta corresponde a resultados obtidos para o *Cadê* com maior número de classes.

A Figura 5.1 mostra uma pesquisa textual efectuada com a expressão de busca "fisica quimica matematica".

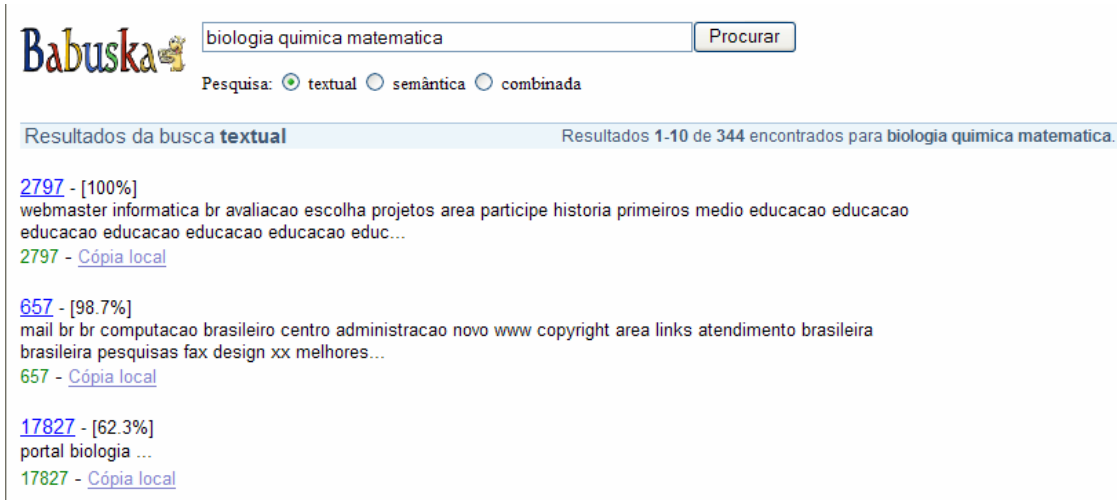

Figura 5.1 – Pesquisa textual efectuada com a expressão de busca "fisica quimica matematica".

Os resultados da busca semântica podem ser observados na Figura 5.2.

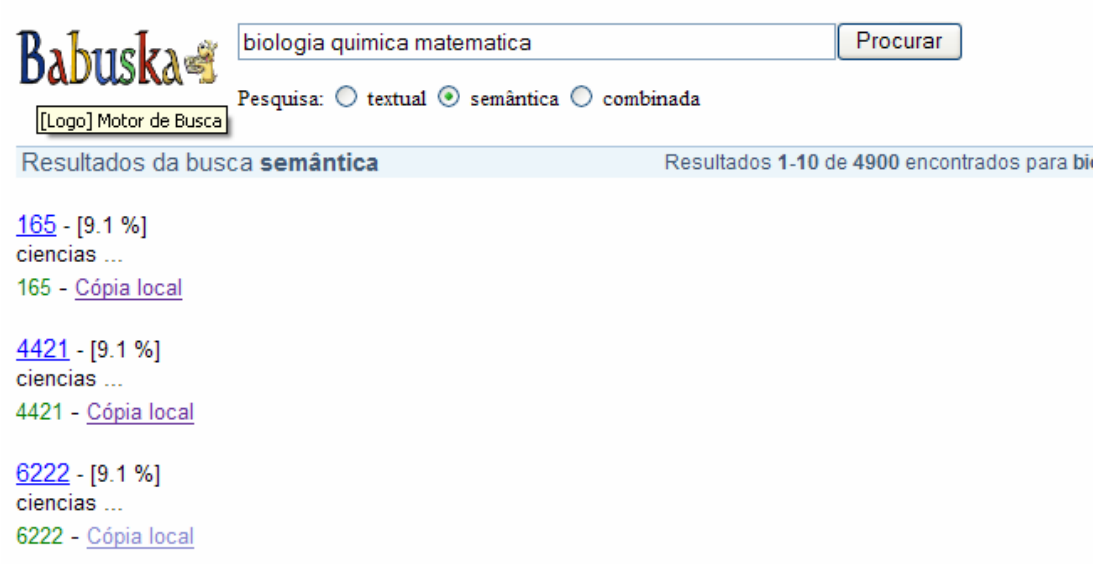

Figura 5.2 – Pesquisa semântica efectuada com a expressão de busca "fisica quimica matematica".

A Figura 5.2 apresenta resultados que têm como classes “ciencias”, “educacao”, entre outras.

Este resultado é bastante elucidativo da validade do método, uma vez que são classes que se esperava ver em resultados semânticos para esta pesquisa.

Com outras expressões de busca os resultados também foram bastante positivos. A expressão “software hardware” dá resultados semânticos pertencentes a classes como “informatica”, “noticiasnotrinf” ou “informaticainfservdiv”. Para a expressão “ingles portugues” as classes dos documentos obtidos na pesquisa semântica são “educacaoedudist”, “educacao”, “educacaoeduling” ou “educacaoeducurso”.

Apesar de bastante positivos os resultados obtidos, o nome das classes, o tipo de documentos da colecção, que são pobres em informação e conteúdo, e a demasiada especificidade dos assuntos tratados nos documentos, limitaram a utilização destas colecções para um estudo mais aprofundado das pesquisas semânticas.

Expressões de busca como “casa branca”, “bola de berlim” ou “white house” não deram sequer resultados semânticos, motivo pelo qual foram criadas outras colecções com mais riqueza de conteúdo. Um dos grandes problemas das colecções da *Reuters* e do *Cadê* corresponde ao facto de terem classes cuja designação não é muito expressiva nem perceptível, o que não permite avaliar convenientemente os resultados.

5.2 Resultados do *Dmoz*

O *Dmoz* foi a primeira solução encontrada para fazer pesquisas com mais significado, uma vez que era possível classificar os documentos de forma mais objectiva (a própria colecção do *Dmoz* tem uma hierarquia bem definida e com classificação de fácil compreensão). Por outro lado, a colecção que se descarregou da *Internet* era fácil de processar e permitia a obtenção das páginas com a ajuda do *Httrack*.

Algumas pesquisas foram efectuadas com os dados recolhidos do *Dmoz*. As expressões de busca utilizadas foram: “casa branca”, “white house”, “desert fox” e “information retrieval” (Figura 5.3).

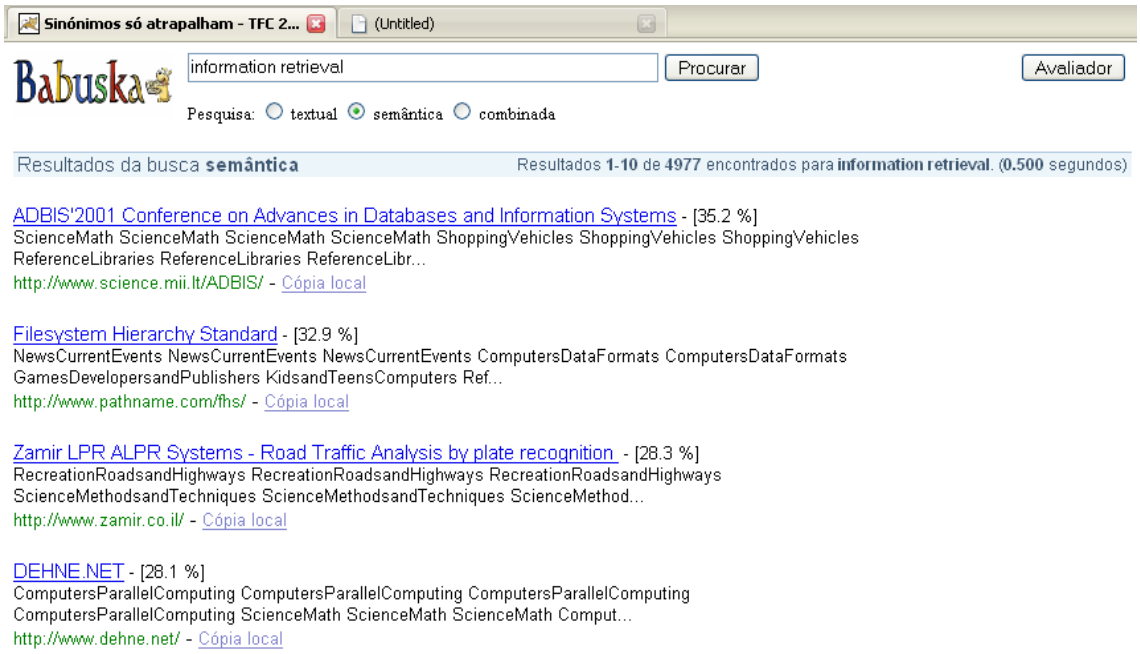

Figura 5.3 – Primeiros resultados da busca semântica para "information retrieval" da colecção do *Dmoz*.

A Figura 5.4 apresenta os resultados em termos de precisão. Pode observar-se facilmente que os resultados são baixos, salientando-se ainda a inexistência de resultados para a expressão "desert fox". O melhor resultado obtido corresponde à expressão "information retrieval" com resultados semelhantes entre as pesquisas combinada e textual, e melhores resultados para a pesquisa semântica.

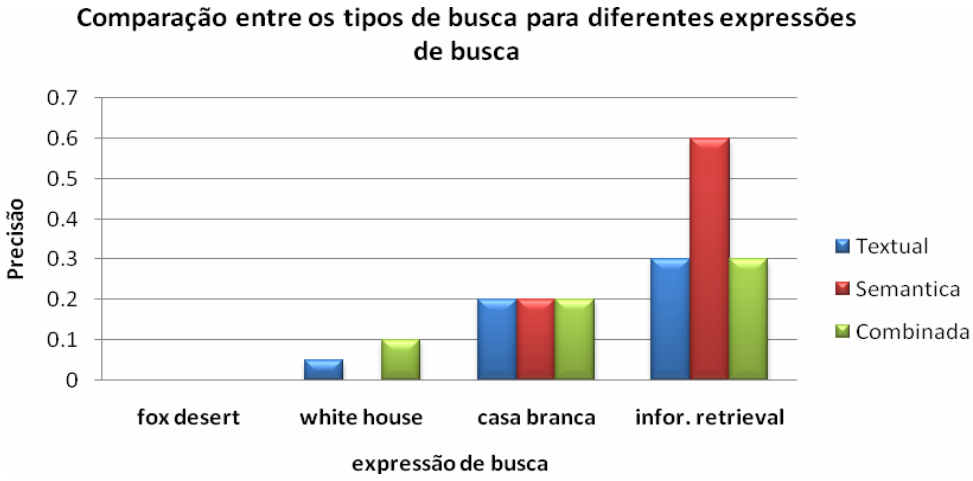

Figura 5.4 – Resultados obtidos na colecção *Dmoz* para diferentes expressões de busca e nos três diferentes tipos de pesquisa (textual, semântica e combinada).

Verifica-se ainda que, para a expressão de busca "white house" os resultados são melhores na pesquisa combinada e inexistentes na semântica. Para a expressão de busca "casa branca" todos os tipos de pesquisa apresentaram o mesmo resultado.

As figuras 5.5, 5.6 e 5.7 apresentam as características das avaliações, nomeadamente do número de avaliadores, resultados relevantes e resultados possíveis de avaliar. Os resultados estão separados nos diferentes tipos de busca suportados pelo Babuska. Nestas figuras, *Avaliadores* corresponde ao número de pessoas que fizeram a avaliação, *Relevantes* corresponde à média do número de resultados avaliados como bons e *Possíveis* corresponde ao número de resultados possíveis de serem avaliados.

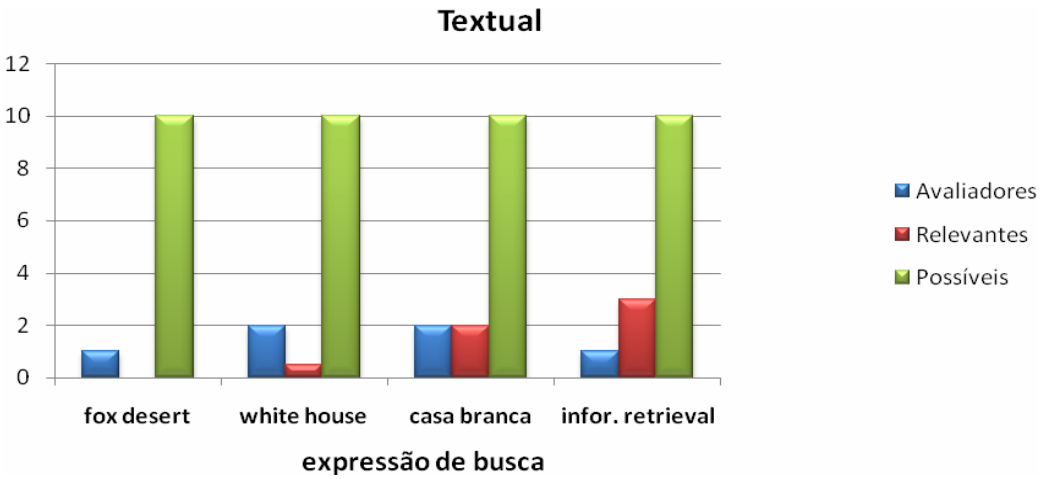

Figura 5.5 – Resultados para a pesquisa textual.

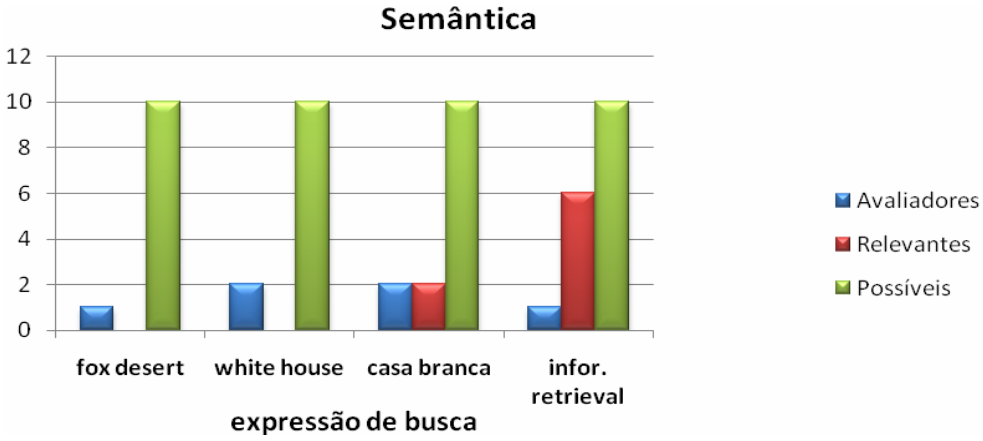

Figura 5.6 – Resultados para a pesquisa semântica.

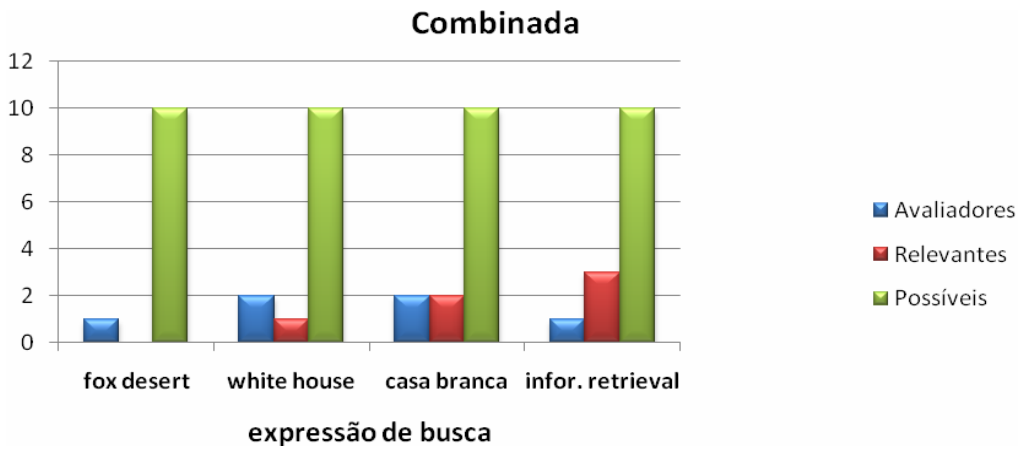

Figura 5.7 – Resultados para a pesquisa combinada.

A colecção do *Dmoz* foi recolhida com apenas um nível de profundidade. Isto significa que apenas foi recolhida a primeira página de cada endereço apontado pelo *Dmoz*. Em muitos casos esta primeira página apenas contém ligações internas, conteúdos em *flash*, páginas que redireccionavam para outras e páginas iniciais meramente de entrada (Figura 5.8). O texto útil obtido é, frequentemente, reduzido. Assim, páginas classificadas como pertencendo a um determinado assunto (classe), acabam por não apresentar conteúdo relevante, dando origem a falsos positivos.

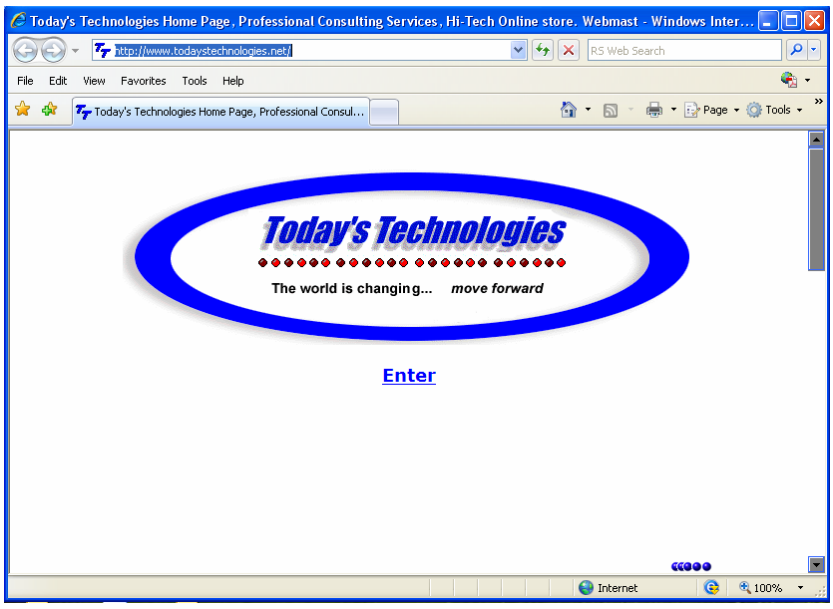

Figura 5.8 – Página inicial sem conteúdo relevante da colecção do *Dmoz*.

Uma vez que o *Dmoz* se encontra hierarquizado com bastante pormenor, e que as páginas apontadas pelo *Dmoz* foram recolhidas em número limitado, a quantidade de documentos pertencentes a cada classe era diminuta pelo que se optou por reduzir a profundidade da categoria atribuída aos documentos. Isto foi feito com o objectivo de conseguir ter mais documentos em cada classe para que cada uma delas fosse mais representativa. Quanto mais refinada for a classificação, mais complicado será estabelecer semelhanças entre classes de documentos. Uma desvantagem deste procedimento reside na redução do número de classes, o que pode provocar uma afinidade generalizada entre os documentos classificados.

5.3 Resultados da *Wikipédia* Portuguesa e Inglesa

De seguida apresentam-se os resultados para as colecções da *Wikipédia* portuguesa e inglesa.

5.3.1 *Wikipédia* Portuguesa

Os resultados obtidos a partir das pesquisas efectuadas à *Wikipédia* portuguesa revelaram-se bastante positivos. Foram escolhidas diferentes expressões de busca consideradas pertinentes para a avaliação dos resultados. Como as colecções anteriores (*Reuters*, *Cadê*, *Dmoz*) eram muito pouco abrangentes em termos de conteúdo as conclusões também foram de certa forma limitadas. Uma das expressões utilizadas na avaliação foi "bola de berlim", que se revelou um bom indicador da qualidade da colecção e do algoritmo.

As Figuras 5.9 e 5.10 mostram os resultados obtidos para diferentes expressões de busca. Fazendo uma análise qualitativa, é perceptível um bom desempenho na pesquisa semântica, melhorando ou mantendo o da pesquisa textual. A pesquisa combinada apresenta, por vezes, piores resultados. Como os resultados desta pesquisa combinam os resultados da pesquisa semântica e textual, os valores obtidos podem dar origem a menos resultados relevantes no final como consequência das combinações das pontuações obtidas pelos documentos nas buscas textual e semântica.

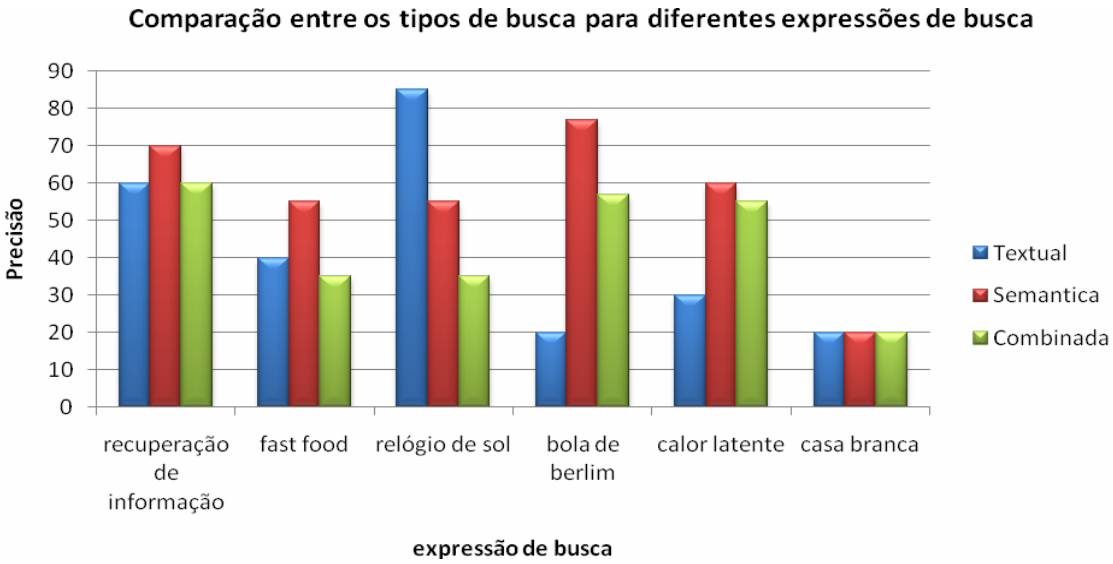

Figura 5.9 – Resultados obtidos na colecção *Wikipédia* portuguesa para diferentes expressões de busca.

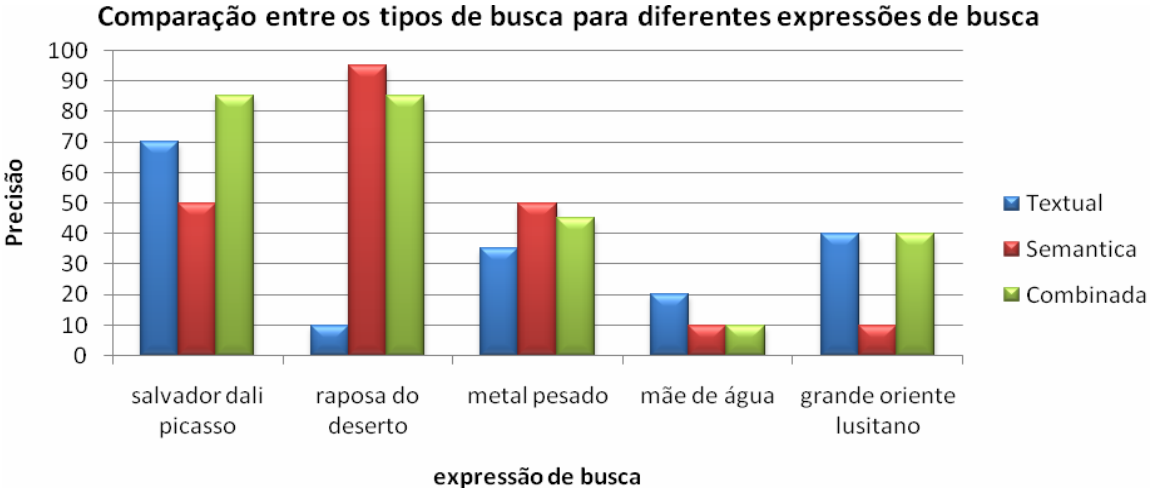

Figura 5.10 – Resultados obtidos na colecção *Wikipédia* portuguesa para diferentes expressões de busca.

De uma forma geral os resultados são bons para qualquer expressão de busca, em qualquer tipo de pesquisa (textual, semântica e combinada). Analisando mais pormenorizadamente os resultados obtidos, verifica-se que "casa branca" e "mãe de água" foram as expressões que deram origem a resultados menos relevantes. Salienta-se o facto de as expressões "raposa do deserto" e "bola de berlim" originarem uma diferença significativa entre as pesquisas textual e semântica, o que é demonstrativo do bom desempenho dos algoritmos desenvolvidos. No

entanto, há expressões de busca com melhores resultados na pesquisa textual do que na pesquisa semântica.

As Figuras 5.11, 5.12 e 5.13 apresentam as características das avaliações, nomeadamente do número de avaliadores, resultados avaliados e resultados possíveis de avaliar. Nestas figuras, *Avaliadores* corresponde ao número de pessoas que fizeram a avaliação e *Relevantes* corresponde à média do número de resultados avaliados como bons.

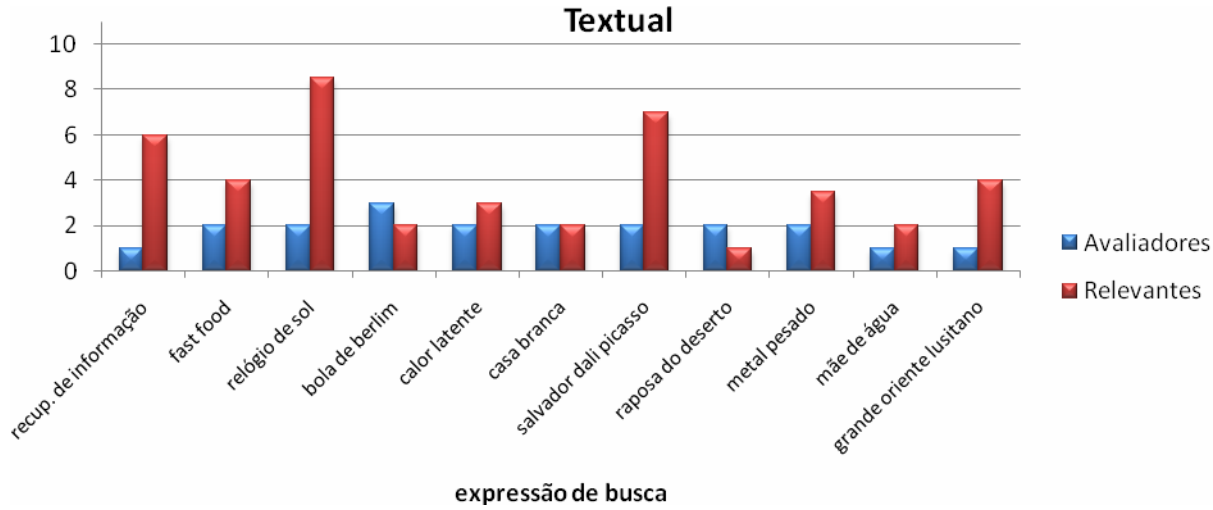

Figura 5.11 – Resultados obtidos para a pesquisa textual.

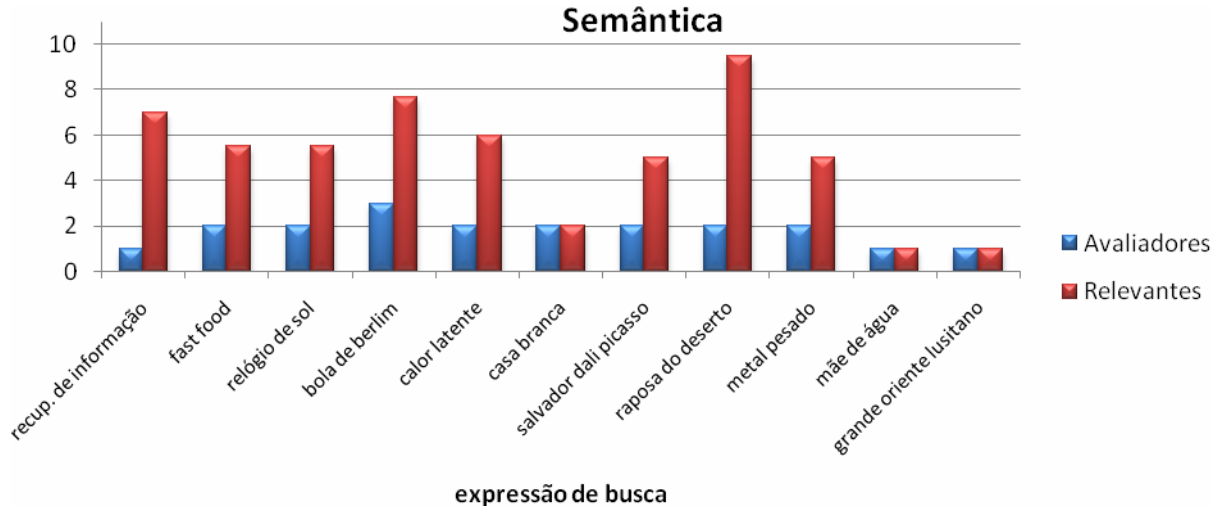

Figura 5.12 – Resultados obtidos para a pesquisa semântica.

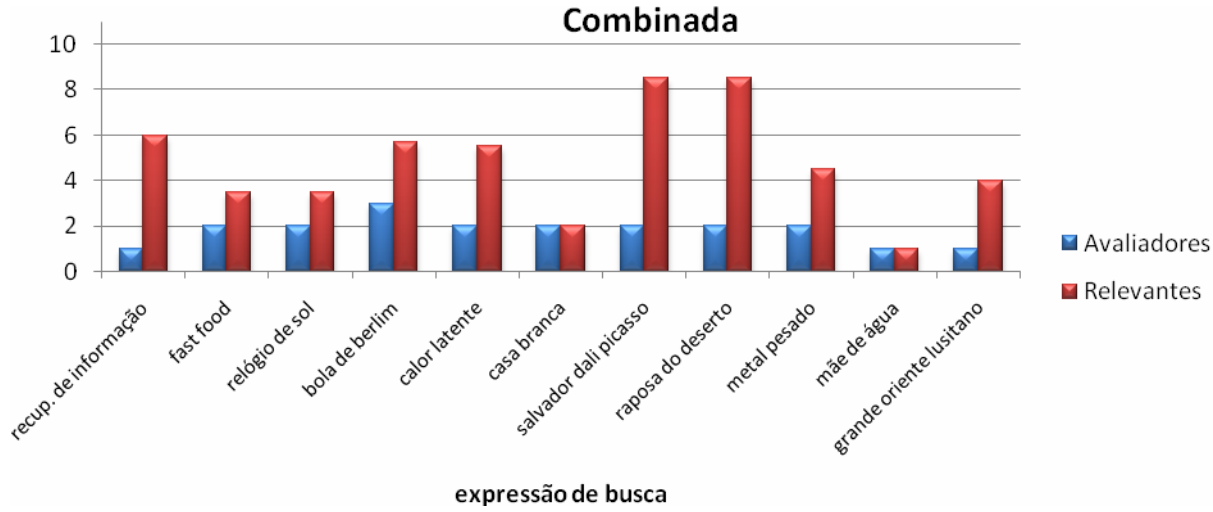

Figura 5.13 – Resultados obtidos para a pesquisa combinada.

Na Figura 5.14 pode observar-se uma pesquisa para a expressão "information retrieval". Como referido anteriormente, os resultados podem ser influenciados consoante as colecções utilizadas e a sua adequabilidade em relação à língua utilizada na expressão de busca. Pesquisar por uma expressão em inglês numa colecção tipicamente em língua portuguesa poderá produzir resultados insatisfatórios, o mesmo sucedendo em situação inversa. Desta forma, podemos contrapor os resultados da Figura 5.14 aos da Figura 5.15, em que se mostra que a pesquisa por "recuperação de informação" deu melhores resultados do que os obtidos para a expressão de busca "information retrieval".

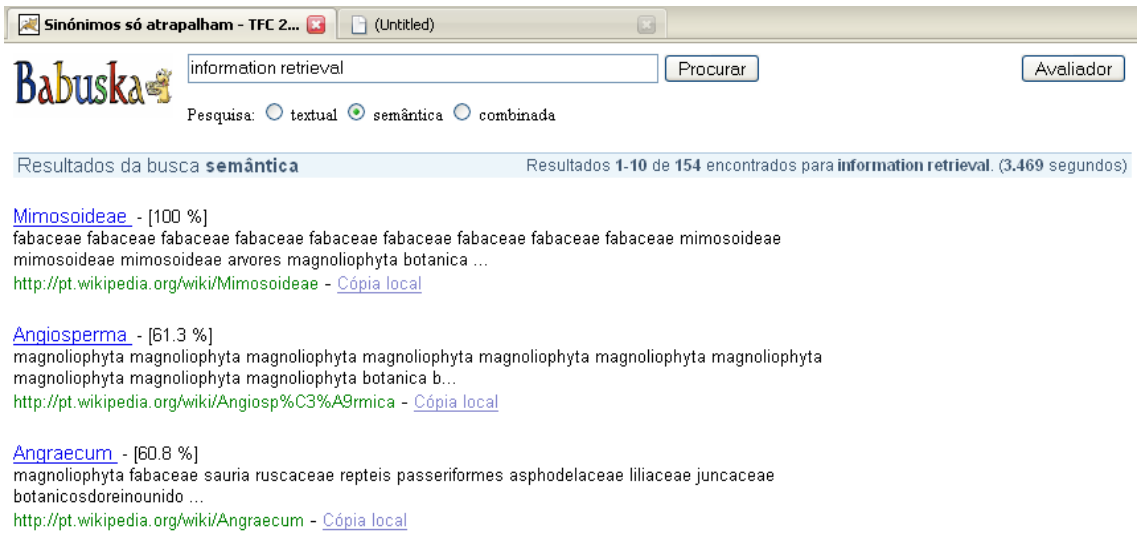

Figura 5.14 – Primeiros resultados da busca semântica para "information retrieval".

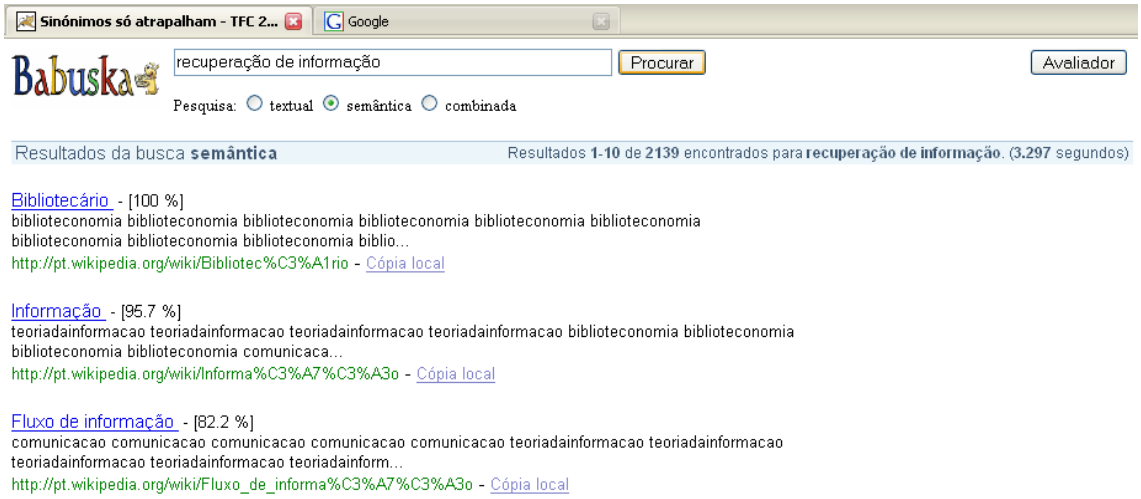

Figura 5.15 – Primeiros resultados da busca semântica para "recuperação de informação".

5.3.2 *Wikipédia* Inglesa

Os resultados das pesquisas efectuadas na *Wikipédia* inglesa encontram-se descritos de seguida. Foram utilizadas algumas expressões de busca, como "white house", "stars stripes", entre outras.

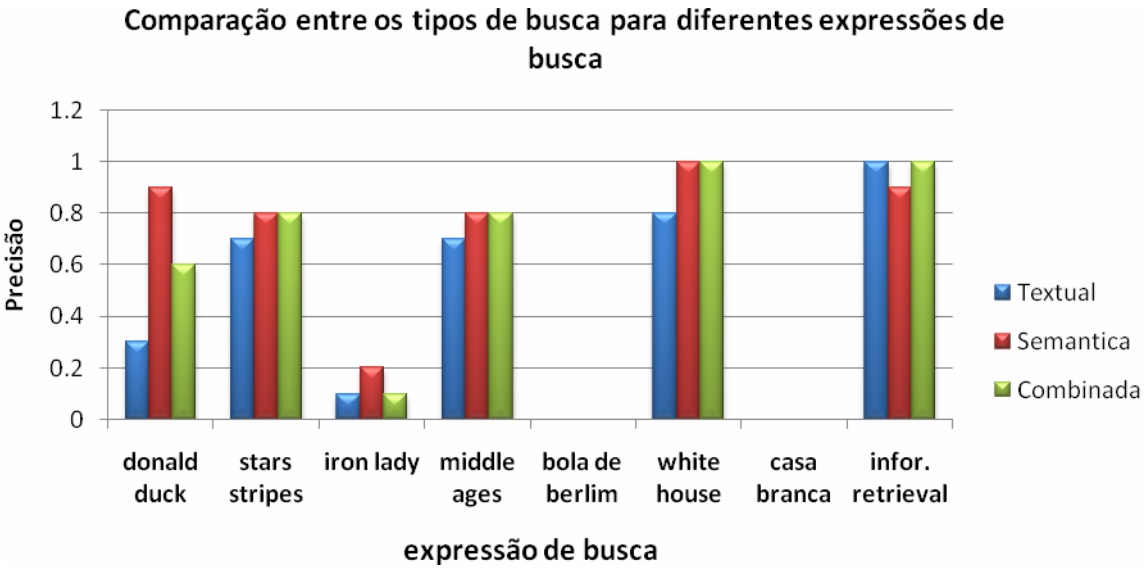

Figura 5.16 – Resultados obtidos na colecção *Wikipédia* inglesa para diferentes expressões de busca.

Observando atentamente a Figura 5.16, verifica-se que não houve quaisquer resultados em duas das sete expressões de busca utilizadas, nomeadamente as expressões de busca "bola de berlim" e "casa branca". Salienta-se a importância deste facto porque as expressões de busca são em português e a colecção utilizada é a *Wikipédia* inglesa.

Para os demais resultados, em que a expressão de busca introduzida é em inglês, os resultados são variados. Os melhores resultados foram obtidos para a expressão "white house" e "information retrieval". Contudo, as diferenças entre os vários tipos de pesquisa são mínimos, o mesmo acontecendo para as expressões de busca "star stripes" e "middle ages". Verifica-se, no entanto, que as pesquisas semânticas e combinadas originam resultados superiores às pesquisas meramente textuais.

As Figuras 5.17, 5.18 e 5.19 apresentam as características das avaliações, nomeadamente do número de avaliadores, resultados avaliados e resultados possíveis de avaliar. Os resultados estão separados nos diferentes tipos de busca suportados pelo Babuska. Nestas figuras, *Avaliadores* corresponde ao número de pessoas que fizeram a avaliação e *Relevantes* corresponde à média do número de resultados avaliados como bons.

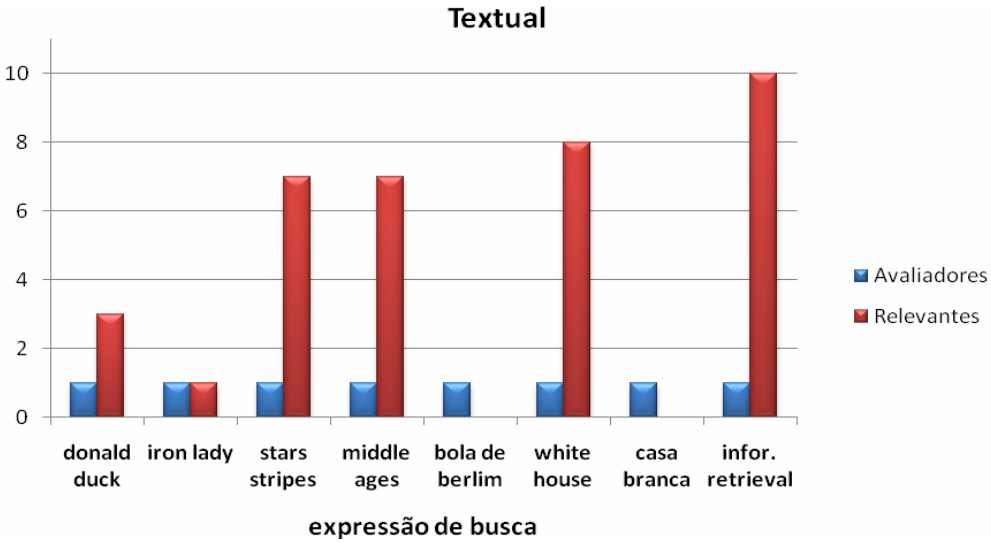

Figura 5.17 – Resultados obtidos para a pesquisa textual.

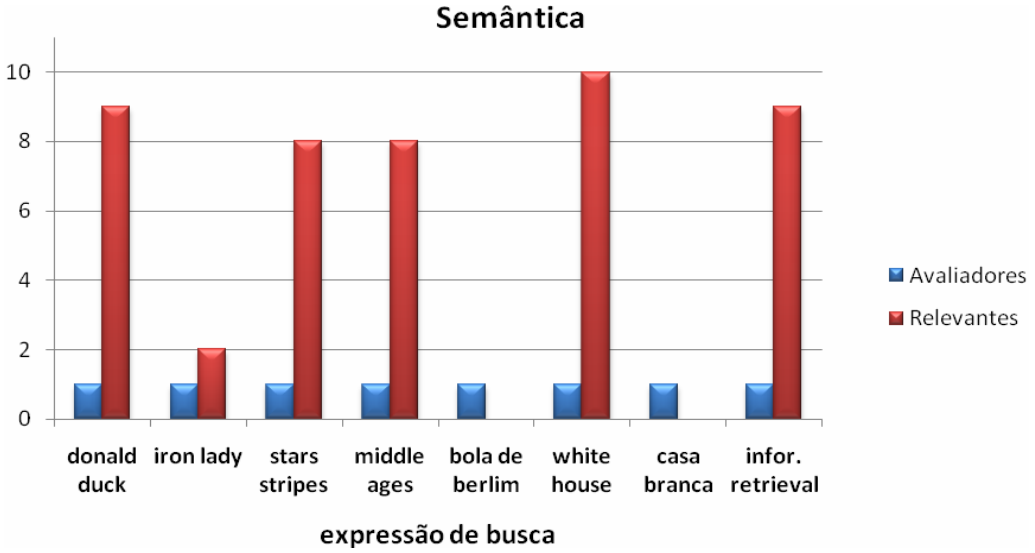

Figura 5.18 – Resultados obtidos para a pesquisa semântica.

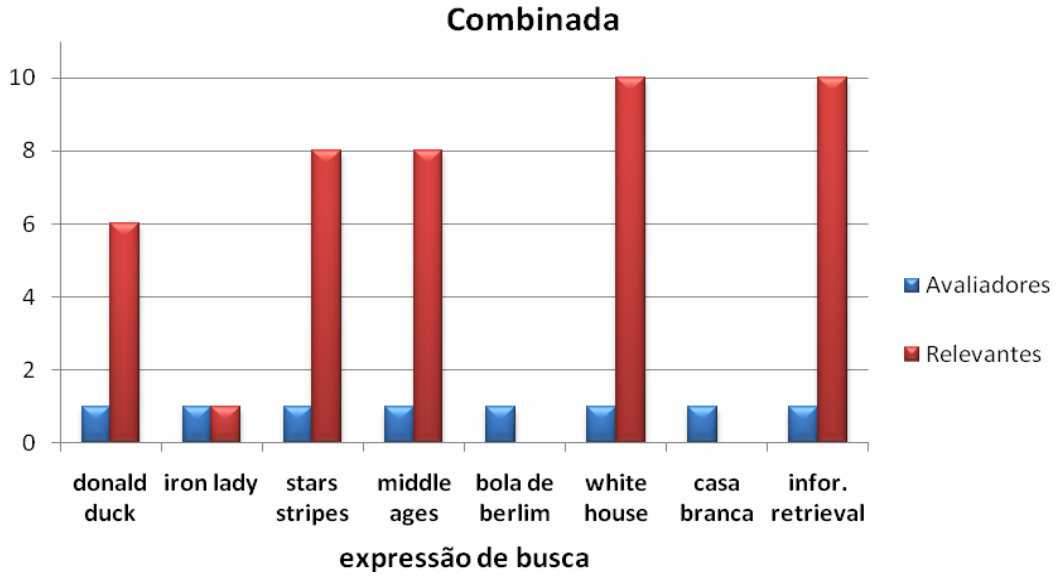

Figura 5.19 – Resultados obtidos para a pesquisa combinada.

5.4 Comparação entre as colecções do *Dmoz* e das *Wikipédias*

A comparação dos resultados das várias colecções pode ser observada nas figuras seguintes. Esta comparação visa dar uma ideia de que as expressões de busca podem originar resultados diferentes para colecções diferentes. Os resultados obtidos mostram que é possível fazer-se

busca semântica em qualquer colecção que se pesquise, mas a relação semântica depende, também, do tipo de colecção utilizada.

A Figura 5.20 apresenta os resultados nas diferentes colecções para a expressão de busca "casa branca". Como se pode observar não foram obtidos quaisquer resultados na colecção da *Wikipédia* inglesa. Para as restantes duas colecções, os resultados são semelhantes. Pode justificar-se a ausência de resultados na *Wikipédia* inglesa com a própria expressão de busca, pois corresponde a uma frase escrita em português.

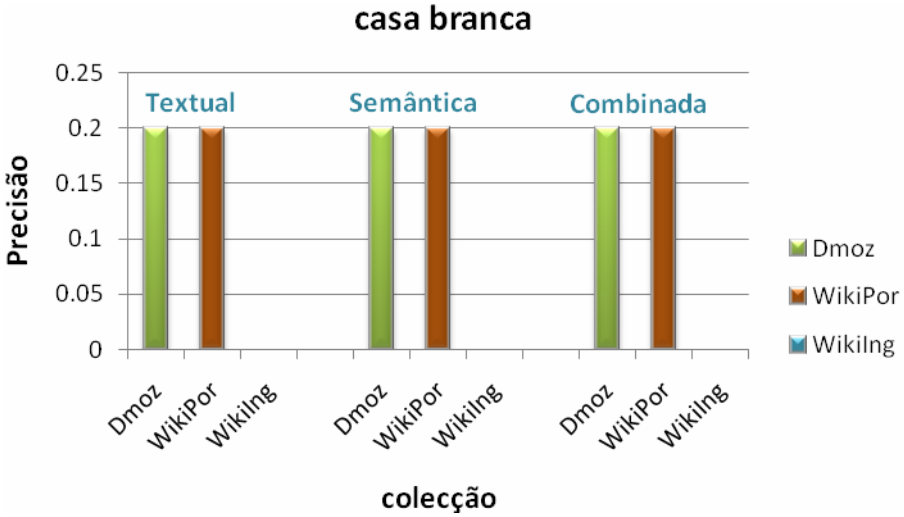

Figura 5.20 – Comparação dos resultados obtidos para a expressão de busca "casa branca" para as diferentes colecções utilizadas (*Dmoz*, *Wikipédia* portuguesa e *Wikipédia* inglesa).

A Figura 5.21 mostra os resultados da expressão de busca "white house" para as colecções *Dmoz* e *Wikipédia* inglesa. Como se pode observar a *Wikipédia* inglesa apresentou melhor resultados do que o *Dmoz*, o que pode ser atribuído às diferenças entre as duas colecções, nomeadamente no que diz respeito ao número de documentos, classes e conteúdo.

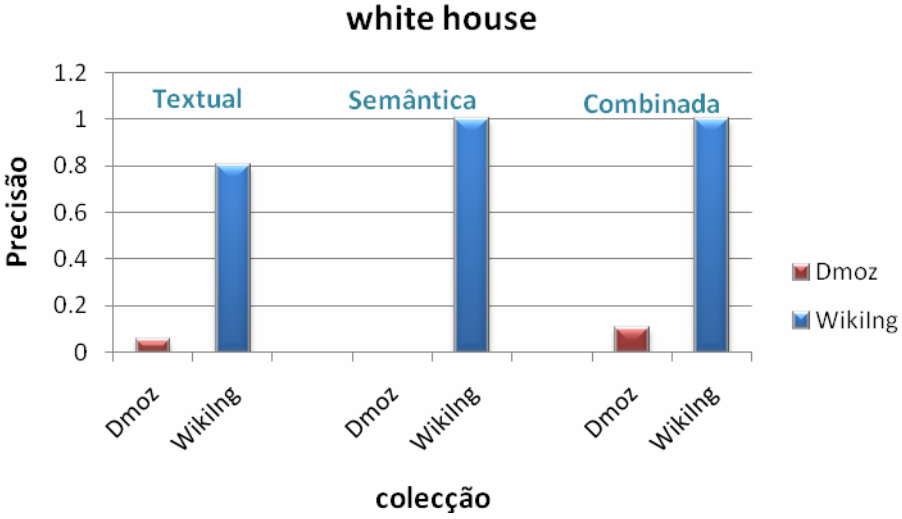

Figura 5.21 – Comparação para a expressão de busca "white house" nas colecções *Wikipédia* inglesa e *Dmoz*.

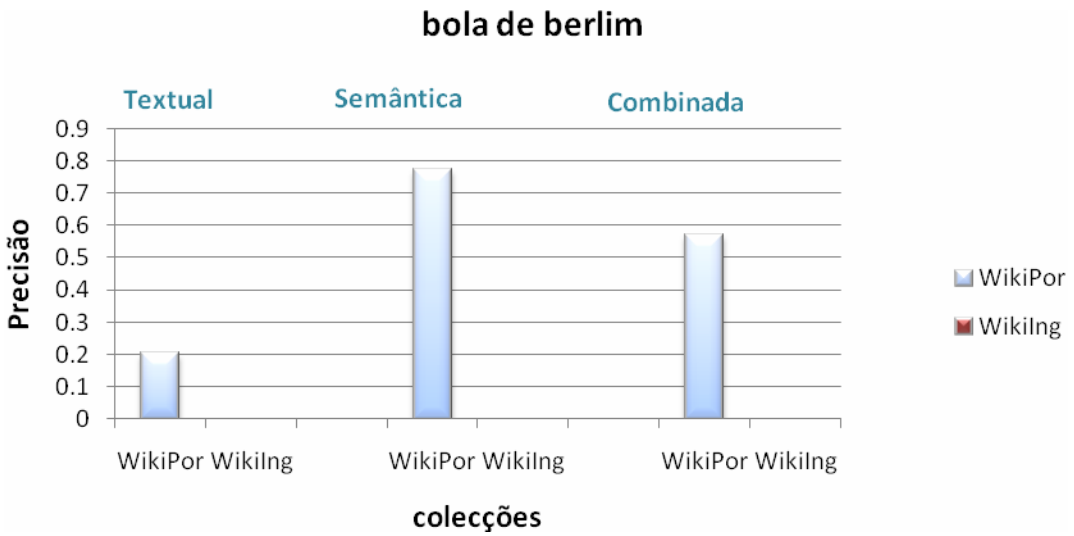

Figura 5.22 - Comparação para a expressão de busca "white house" nas colecções *Wikipédia* inglesa e *Wikipédia* portuguesa.

Comparou-se também as *Wikipédias* inglesa e portuguesa na expressão de busca "bola de berlim", cujos resultados se podem observar na Figura 5.22. Tal como foi referido para os resultados da expressão de busca "casa branca", aqui não se obtiveram resultados para a *Wikipédia* inglesa. A explicação é provavelmente a mesma que foi dada para o caso da expressão "casa branca".

Finalmente, apresentam-se os resultados para uma combinação das expressões de busca "information retrieval" e "recuperação de informação". A distinção entre estas duas formas de pesquisa reside na diferença das colecções. O objectivo corresponde a verificar se para expressões equivalentes, neste caso com o mesmo significado mas em línguas diferentes, podem ser produzidos resultados semelhantes. Observando atentamente a Figura 5.23 é possível observar que todas as colecções produziram resultados em todos os tipos de busca (textual, semântica e combinada). Verifica-se ainda que a pesquisa da *Wikipédia* inglesa é a que melhores resultados origina. Contudo, para o caso da *Wikipédia* portuguesa os resultados também foram bons. Salienta-se ainda que os resultados da pesquisa semântica foram bastante homogéneos entre as três colecções.

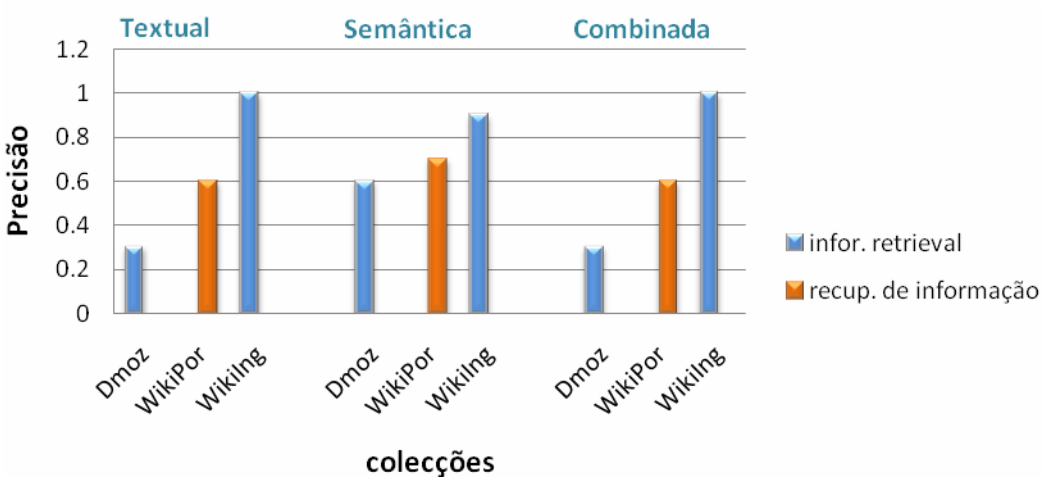

Figura 5.23 - Comparação para a expressão de busca "information retrieval" e "recuperação de informação" nas colecções *Wikipédia* inglesa e *Wikipédia* portuguesa.

5.5 Comparação entre o algoritmo inicial e final utilizando a *Wikipédia* portuguesa

É possível verificar diferenças entre o algoritmo desenvolvido inicialmente e o algoritmo utilizado actualmente no Babuska, tal como descrito na secção 4.4.3. As Figuras 5.24 e 5.25 apresentam um exemplo das diferenças observadas utilizando "bola de berlim" como expressão de busca. A pesquisa semântica obtida no primeiro caso (Figura 5.24) apresenta resultados aparentemente fracos. Numa análise mais atenta à figura verifica-se que os resultados têm pouca afinidade com a expressão de busca. Outro aspecto a salientar é o

número de resultados obtidos, 18579 documentos, ou seja, a maioria da colecção, o que significa que, isoladamente, as palavras "bola" e "berlim" teriam, provavelmente, afinidade com a maioria das classes. A pesquisa semântica obtida para o segundo caso (Figura 5.25) mostra um comportamento completamente diferente, a começar pelo número de resultados obtidos, 1900 documentos. Este número mostra um grande refinamento na procura. Contudo, a diferença mais notória, e a mais relevante, verifica-se, sem dúvida, nos resultados obtidos. Como se pode observar na Figura 5.25, estes estão relacionados essencialmente com culinária ou bolos.

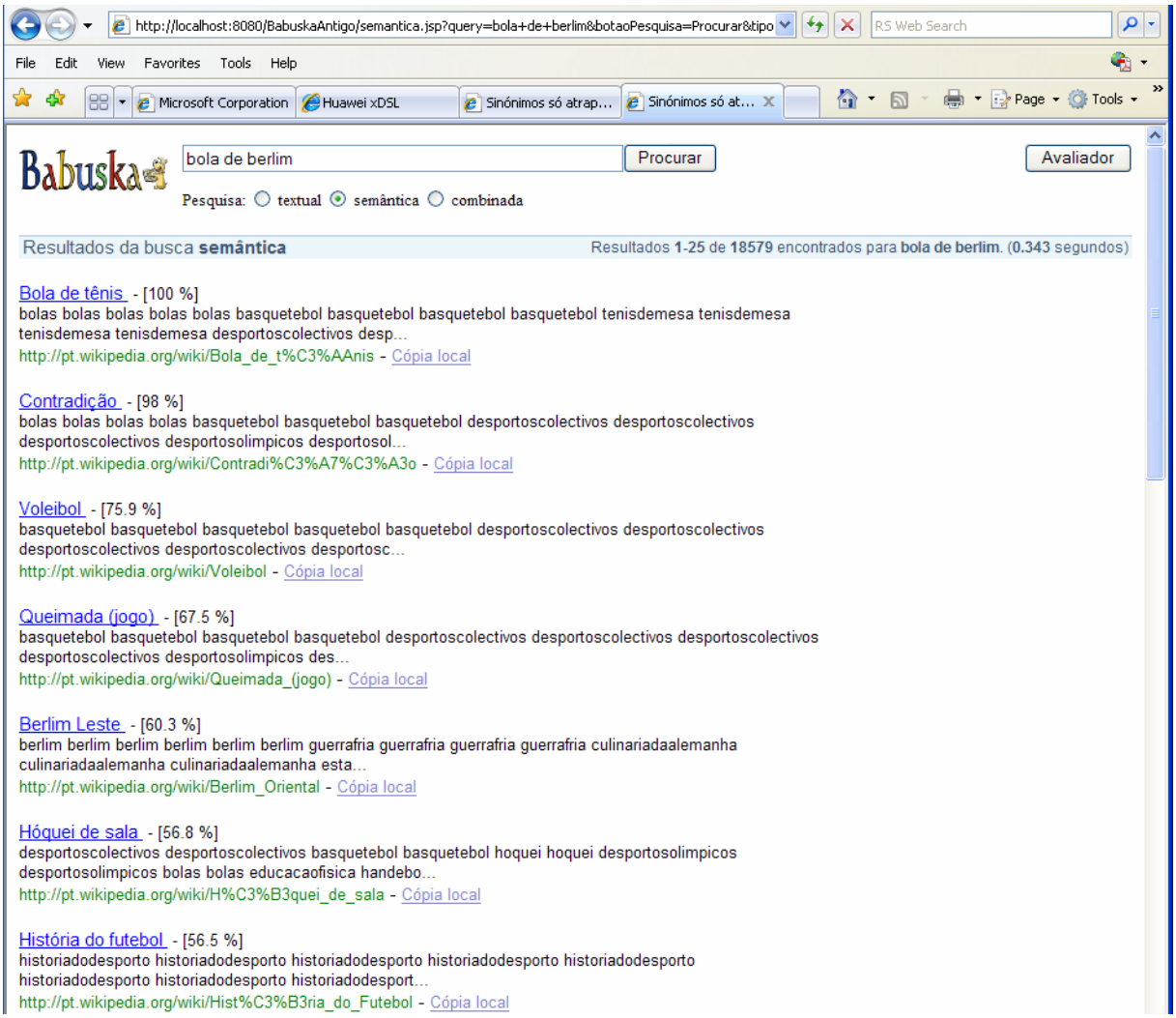

Figura 5.24 – Resultados obtidos para a expressão de busca "bola de berlim" com o algoritmo de transformação da expressão inicialmente desenvolvido.

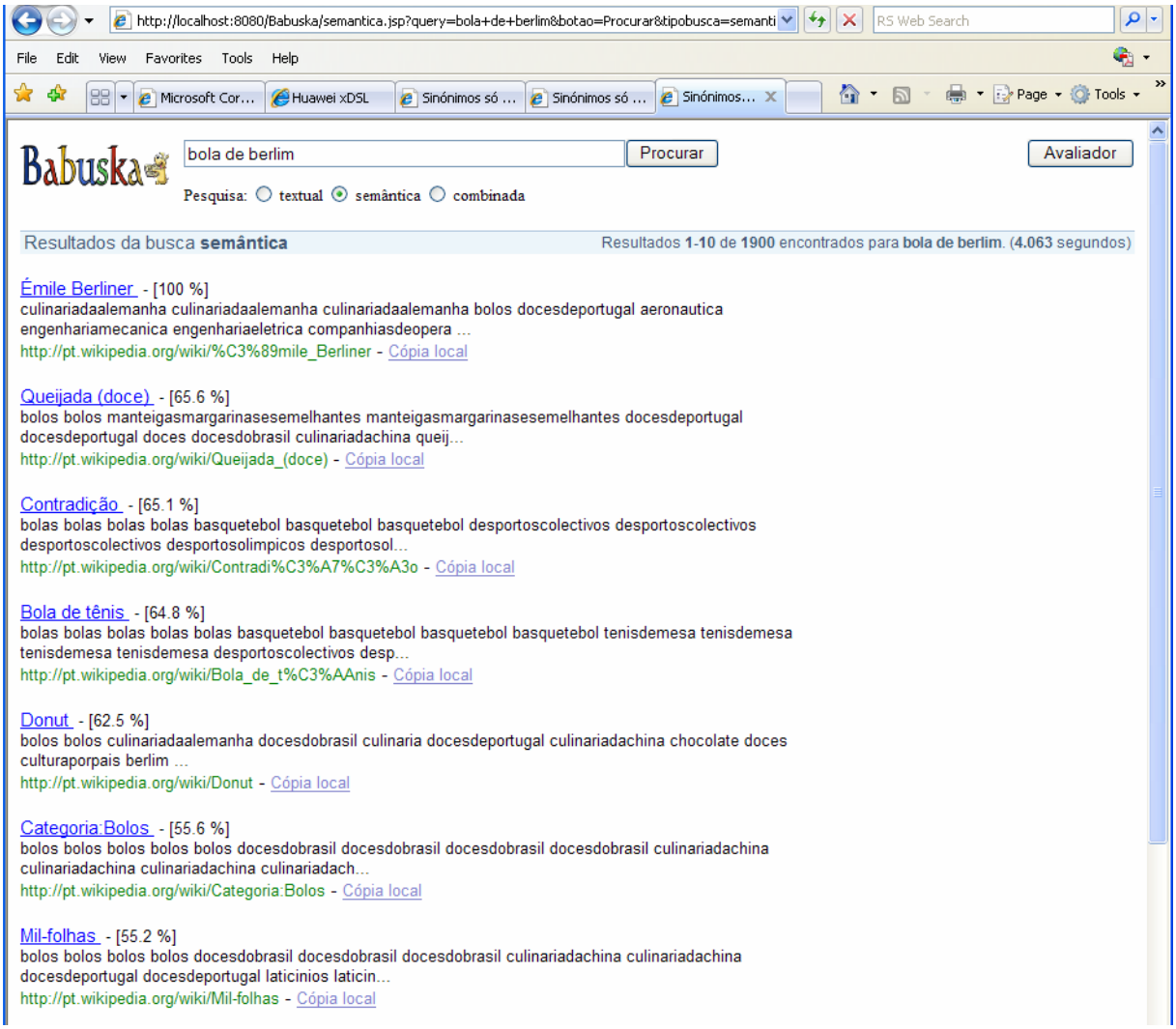

Figura 5.25 – Resultados obtidos para a expressão de busca "bola de berlim" com o algoritmo final de transformação da expressão.

5.6 Análise às diferentes grafias

Importa também analisar um outro aspecto muito relevante nas pesquisas efectuadas. É necessário ter em consideração como se introduzem as expressões de busca. Não se trata do simples corrigir de erros ortográficos, como por exemplo, "bola de berlmi" em vez de "bola de berlim", mas da forma como certas palavras são escritas para o mesmo idioma em países diferentes. Um dos casos mais evidentes é o caso do português de Portugal e o português do Brasil onde muitas palavras são escritas de forma diferente. Um destes exemplos é o caso do termo "ião", para o português de Portugal e "iôn" para o português do Brasil. Os resultados que se obtêm são diferentes consoante se use um termo ou outro.

Um exemplo a explorar corresponde à expressão "ião cloreto" e "iôn cloreto". As páginas idênticas para as duas pesquisas correspondem às classes que contêm ambos os termos. A Figura 5.26 corresponde à busca textual de "ião cloreto" e a Figura 5.27 à expressão "iôn cloreto". Como se pode verificar, as buscas textuais originam resultados bastante diferentes.

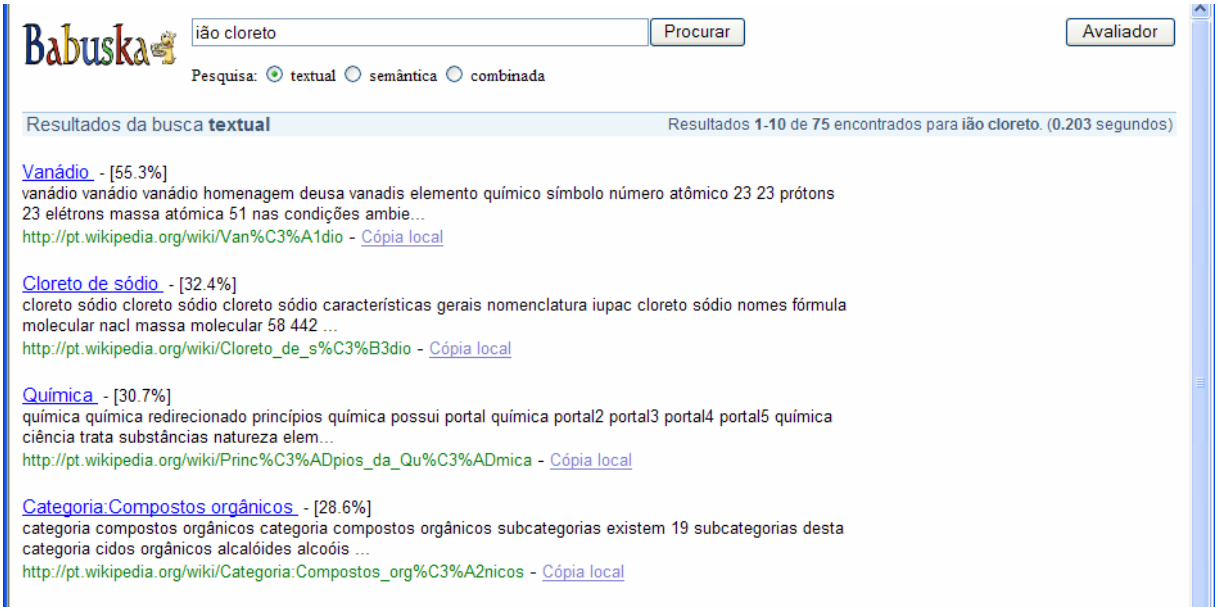

Figura 5.26 - Busca textual de "ião cloreto".

Figura 5.27 - Busca textual de "iôn cloreto".

Em seguida apresenta-se qual o comportamento semântico das duas expressões (Figuras 5.28 e 5.29).

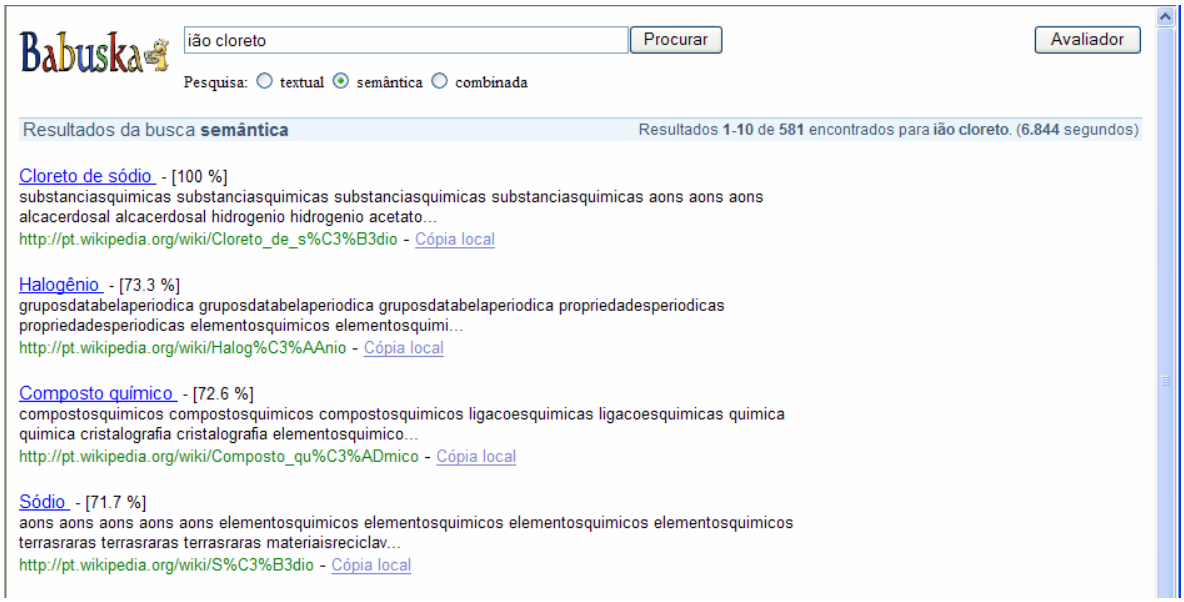

Figura 5.28 - Busca semântica de "ião cloreto".

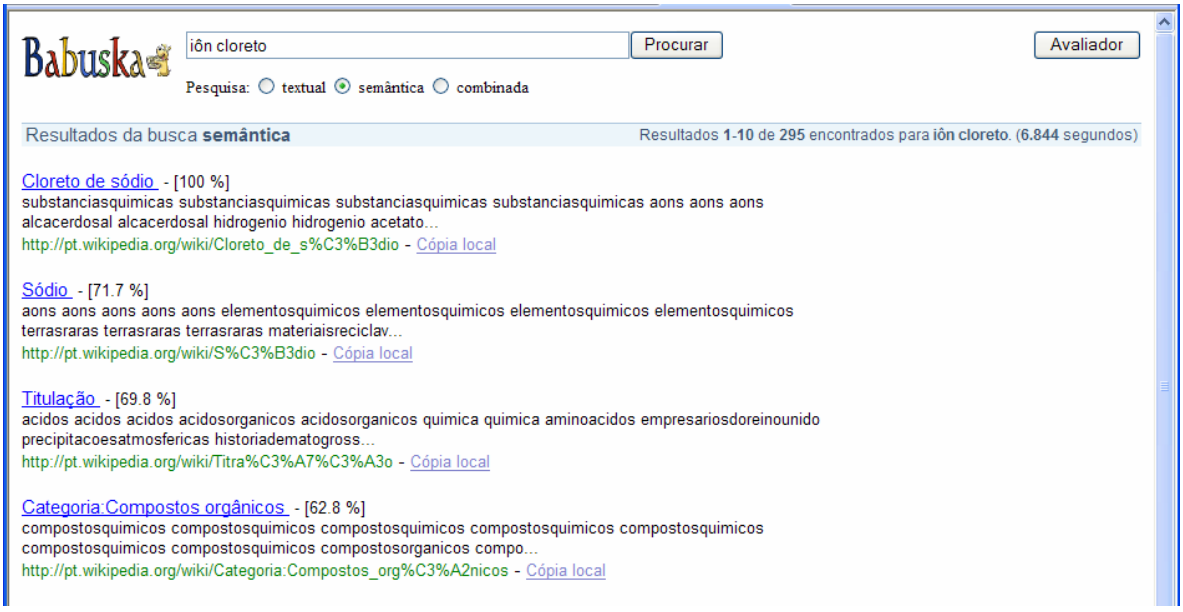

Figura 5.29 - Busca semântica de "iôn cloreto".

Os resultados obtidos semanticamente mostram que seria desejável ultrapassar este problema, pois existem documentos que se poderiam considerar bons tanto numa pesquisa como noutra, já que ambas são a mesma pesquisa só que com grafia diferente.

A título de exemplo, a busca semântica numa combinação de "ião iôn cloreto" originaria o resultado que se observa na Figura 5.30.

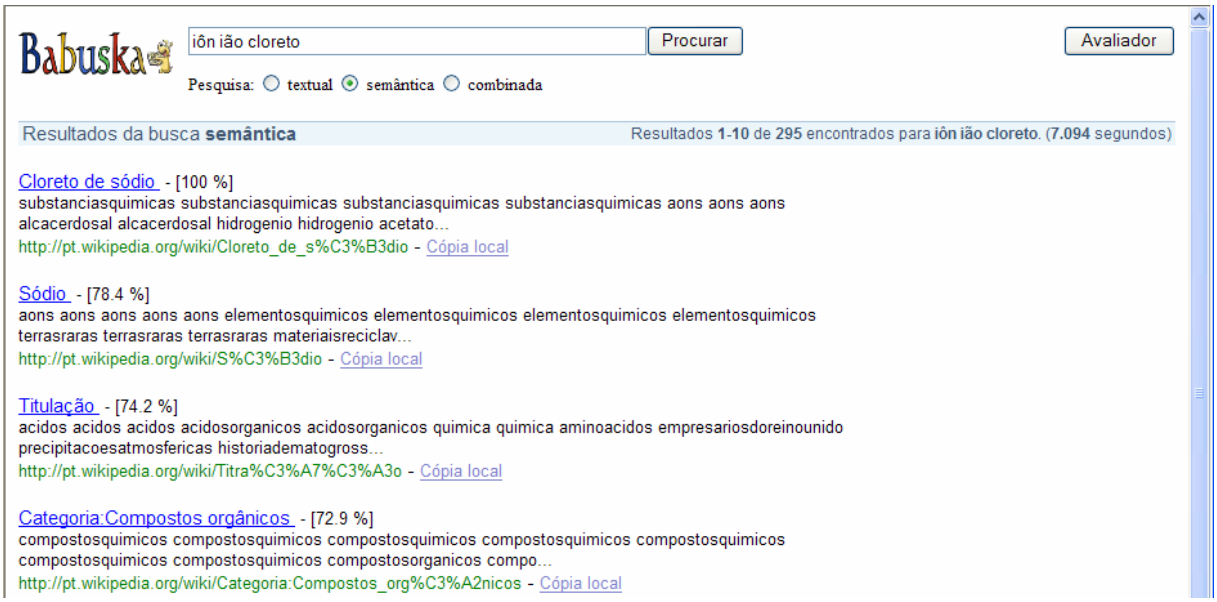

Figura 5.30 – Pesquisa semântica para a expressão de busca "iôn ião cloreto".

A Figura 5.30 mostra outro aspecto interessante: o resultado é o mesmo da pesquisa semântica da expressão "iôn cloreto". Este resultado mostra que a palavra "ião" pouco significado tem para a pesquisa, ou seja, não tem importância suficiente para alterar o panorama dos resultados para próximo dos que se obtiveram quando da pesquisa por "ião cloreto". Verifica-se a alteração nos valores de pontuação obtidos nos resultados das figuras 5.29 e 5.30, devido à introdução do termo ião na expressão de busca.

6. Conclusões

O trabalho foi desenvolvido e concluído conforme o planeamento previsto. Foram criadas diversas colecções e desenvolvidos algoritmos para a transformação de documentos. O sistema foi materializado num motor de busca.

As várias colecções que foram utilizadas para a realização do trabalho mostraram diferentes comportamentos e, consequentemente, os seus resultados também foram diferentes. Estas diferenças foram importantes para a avaliação do comportamento do todo o sistema. As colecções da *Reuters* e do *Cadê* permitiram dar início ao desenvolvimento do sistema. O uso destas colecções foi muito importante na fase de desenvolvimento dos algoritmos porque permitiu testar a sua aplicabilidade e a viabilidade das pesquisas semânticas.

Um dos problemas que as colecções da *Reuters* e do *Cadê* levantaram (pouca perceptibilidade do significado das classes) levou à procura de colecções com conteúdo mais fácil de trabalhar e de avaliar. Desta forma, foi criada inicialmente a colecção do *Dmoz*. Os resultados desta colecção ficaram aquém do esperado, em virtude de ter poucas classes, pouca profundidade na obtenção das páginas através do *Httrack*, ficando alguns documentos reduzidos a uma página inicial irrelevante, documentos com pouca informação (texto) e fraca representatividade dos documentos e das classes.

Os resultados para a *Wikipédia* portuguesa foram melhores com as expressões de busca em português. Como se pode verificar no capítulo 5, não se obtiveram resultados satisfatórios para a expressão "information retrieval" ao contrário do que foi obtido com a expressão "recuperação de informação".

Grafias diferentes da mesma palavra, consoante as variedades linguísticas (por exemplo, português de Portugal e português do Brasil), influenciam os resultados obtidos nas pesquisas efectuadas, tal como observado na análise de resultados. No entanto, a pesquisa semântica poderá evitar estes problemas pois as palavras ficam muitas vezes associadas a classes às quais pertencem as diferentes palavras. De modo a aferir com mais certeza a melhoria que a pesquisa semântica introduz seria necessário explorar exaustivamente esta questão.

Efectivamente, houve melhorias aos resultados das pesquisas textuais, aplicando a pesquisa semântica e combinada. Por exemplo, na procura por "bola de berlim" os resultados

relacionados com desporto e com a capital da Alemanha foram desaparecendo dos primeiros lugares ou virtualmente eliminados na pesquisa semântica, onde a maioria dos resultados avaliáveis estava relacionada com culinária ou bolos.

A evolução do algoritmo de transformação da expressão de busca (como descrito na secção 4.4.3) permitiu melhorar os resultados obtidos para as pesquisas semânticas.

A agregação dos documentos pertencentes à mesma classe pode dar origem a resultados inesperados (secção 4.4.1). Isto acontece devido à possibilidade de se estarem a relacionar documentos que, apesar de pertencerem à mesma classe, digam respeito, efectivamente, a matérias diferentes. Este problema foi observado na pesquisa da expressão "information retrieval" na *Wikipédia* portuguesa, onde apareciam resultados de flores cujos documentos continham esta expressão. A existência desta expressão foi condição suficiente para obter uma afinidade com uma classe com a qual não tinham evidente relação.

O cálculo das afinidades está dependente das normas dos vectores. Quanto maior for a norma, menor é a afinidade. Isto leva, por vezes, a uma diminuição das afinidades obtidas se os documentos forem demasiado grandes, reduzindo a importância de uma classe quando tal não deveria acontecer.

O número de classes entre as colecções era bastante variado, influenciando os resultados obtidos. Não foi possível aferir se as 17 mil classes da *Wikipédia* inglesa influenciavam positivamente os resultados em relação às 7 mil classes da *Wikipédia* portuguesa.

Não é possível avaliar com exactidão de que forma o número de classes influencia os resultados. Um outro aspecto importante que não foi possível avaliar em tempo útil corresponde à análise do comportamento do sistema utilizando diferentes tamanhos dos conjuntos de treino e de teste.

7. Trabalho futuro

A avaliação dos resultados foi efectuada pelos elementos do grupo. Para um trabalho mais científico seria necessário ter um conjunto de amostragem superior de resultados avaliados e elementos avaliadores.

A optimização dos parâmetros é outro factor fundamental para melhorar o trabalho. Deste modo, mais combinações teriam de ser equacionadas, incluindo aprendizagem. Isto implica a necessidade de alteração dos algoritmos para incorporar métodos de avaliação e classificação dos resultados. Por exemplo, um documento que nunca é escolhido como bom teria a sua pontuação reduzida.

Poderiam ser combinadas várias colecções com o objectivo de conseguir uma única colecção com mais classes e documentos. Teria como vantagem a constituição de uma colecção mais rica que permitiria obter resultados de documentos pertencentes a contextos mais variados.

Poderia ser feita a separação categorizada dos resultados. No caso de existirem vários resultados muito bons mas pertencentes a categorias distintas poderia ser feita uma separação dos resultados tendo em conta as classes a que pertencessem. Esta separação poderia ser efectuada, por exemplo, dividindo a página em três zonas com as três principais categorias encontradas, ou mediante a escolha por parte do utilizador dos resultados a visualizar a partir de uma caixa de selecção de categorias encontradas.

É necessário desenvolver métodos que permitam resolver o problema da variabilidade das línguas, como é o caso dos termos ião (português de Portugal) e iôn (português do Brasil), possibilitando a procura de documentos indiferentemente da grafia dos termos.

O objectivo final do trabalho foi a criação de um motor de busca simples que permitisse observar os resultados dos algoritmos desenvolvidos. O motor poderia ser melhorado com outras funcionalidades, seguindo as melhores práticas disponíveis. Por exemplo, reconhecimento da língua em que a expressão de busca foi escrita e possibilidade de pesquisar a mesma expressão em várias línguas e em várias colecções, sugestões de correcção à expressão de busca inserida pelo utilizador, entre outras funcionalidades possíveis. A eficiência do motor de busca desenvolvido poderia ser melhorada, tornando mais rápida a geração e apresentação dos resultados.

8. Referências

[1] G. Salton, A. Wong, and C. S. Yang, "A Vector Space Model for Automatic Indexing", Communications of the ACM, 1995, vol. 18, nr. 11, pp. 613–620

[2] J. Zobel, A. Moffat, K. Ramamohanarao, “Inverted files versus signature files for text indexing”. Transactions on Database Systems, 23(4), December 1998, pp. 453-490

[3] Becker, J., Kuropka, D., “Topic-based vector space model. Proceedings of the 6th International Conference on Business Information Systems”, Colorado Springs, June 2003, pp. 7-12

[4] http://www.gia.ist.utl.pt/~acardoso/datasets/, em 10-10-2007

[5] http://en.wikipedia.org/wiki/Apache_Tomcat, em 10-10-2007

[6] http://lucene.apache.org/java/docs/, em 10-10-2007

[7] http://www.lucenebook.com/, em 10-10-2007

[8] http://lucene.apache.org/java/docs/queryparsersyntax.html, em 10-10-2007

[9] http://en.wikipedia.org/wiki/Vector_space_model, em 10-10-2007

[10] http://en.wikipedia.org/wiki/Index_(publishing), em 10-10-2007

[11] http://www.httrack.com/, em 10-10-2007

[12] http://www.wikipedia.org, em 10-10-2007

[13] http://rdf.dmoz.org/rdf/content.rdf.u8.gz, em 10-10-2007

[14] http://www.ranks.nl/stopwords/, em 10-10-2007

[15] Ian H. Witten, Alistair Moffat, Timothy C. Bell, “Managing Gigabytes: Compressing and Indexing Documents and Images”, Second Edition Morgan Kaufmann, 1999

[16] Norbert Fuhr, “Probabilistic models in information retrieval”, The Computer Journal, June 1992, vol. 35, nr. 3, pp.243-255

Anexos

Anexo A – Manual para a criação dos ficheiros de classes e afinidades

A aplicação do algoritmo que permite processar os documentos de treino e de teste e obter os documentos indispensáveis para o correcto funcionamento do Babuska é feito usando o programa descrito nesta secção. O programa seguinte permite obter os ficheiros de classes (ficheiro que contém as classes que vão ser usadas para calcular as afinidades) e o ficheiro de afinidades entre o treino e o teste (onde ser irão fazer as procuras). Para o caso de processamento de documentos que tenham ligações para páginas da *Internet* obtém-se ainda um ficheiro contendo a relação entre os ficheiros de treino processados e os endereços web correspondentes.

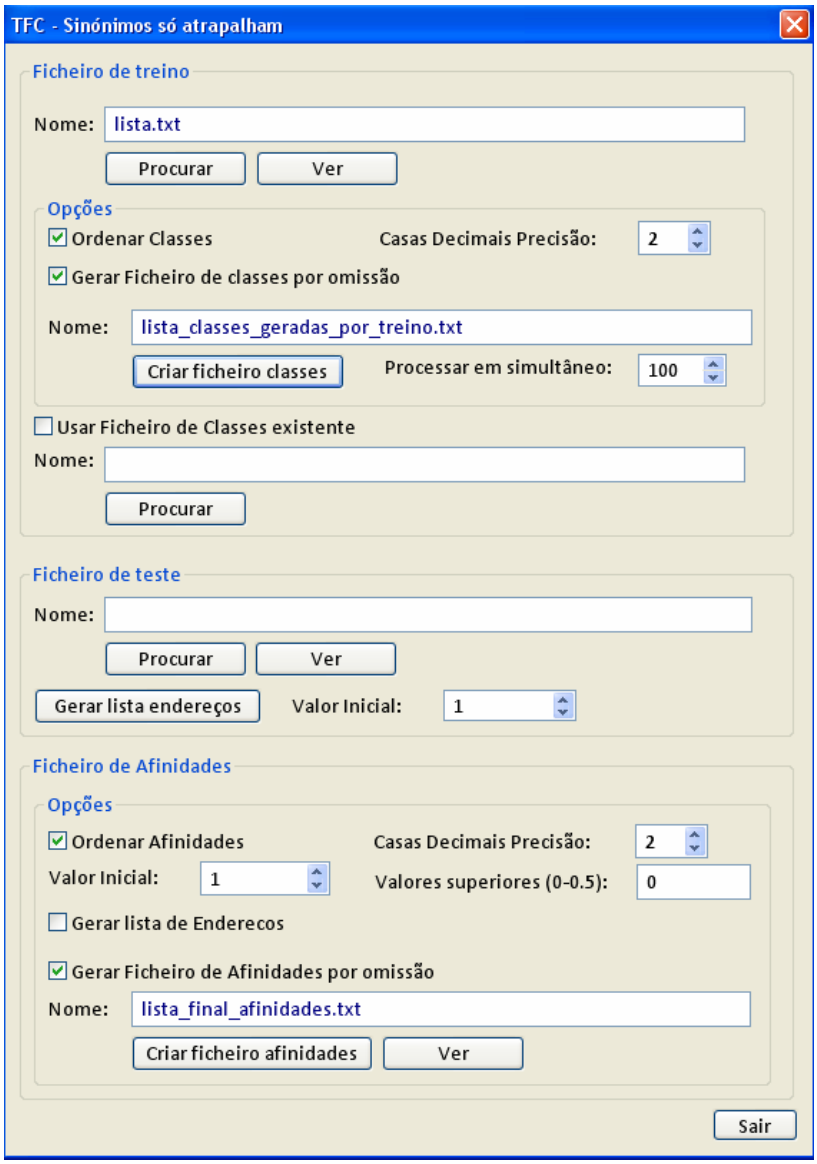

Figura A.1 – Interface gráfica para interacção com o utilizador.

A Figura A.1 corresponde à interface inicial do programa onde todas as opções são escolhidas de modo a obter-se os ficheiros pretendidos. O funcionamento do programa pode ser dividido em diferentes partes, como descrito nas secções seguintes.

A.1 Criar o ficheiro com as classes

Esta opção permite que apenas o ficheiro de classes seja processado. Este ficheiro é o que vai ser usado posteriormente pelo Babuska para que sejam calculadas as afinidades com a *query* de pesquisa no Babuska. Este ficheiro é obtido a partir do ficheiro de treino.

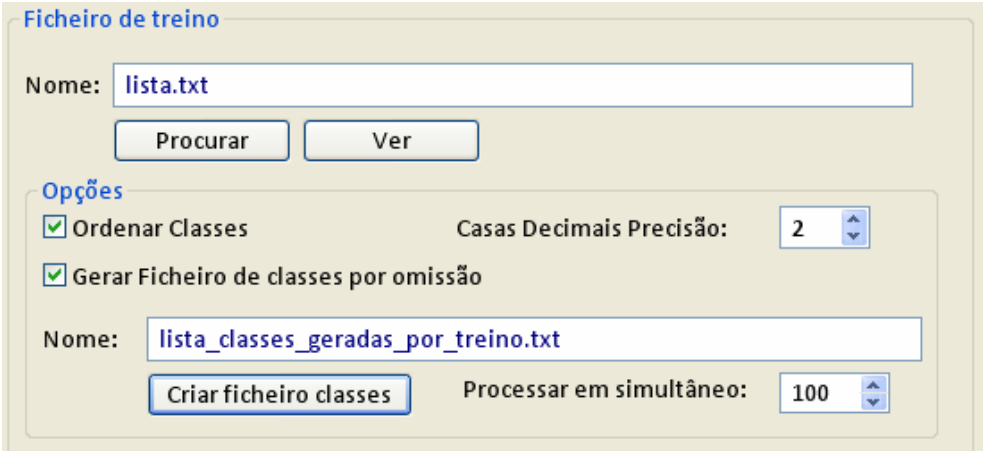

Figura A.2 – parte da interface da Figura A.1 que corresponde à parte onde se pode criar o ficheiro de classes.

Para se criar o ficheiro de classes é necessário seguir os seguintes passos:

1. Escolher o ficheiro de treino a processar ao permir o botão **Procurar** localizado pela caixa de texto antes de **Opções** e no lado esquerdo do botão **Ver**.
2. A caixa **Opções** permite que o ficheiro a gerar tenha as seguintes características:
    a. **Ordenar Classes** – Permite ordenar cada documento de uma classe por ordem decrescente da média do número de palavras que compõem essa classe.
    b. **Casas Decimais de precisão** – Esta opção permite definir quantas casas decimais serão contabilizadas para o valor da média do número de palavras.
    c. **Gerar Ficheiro de classes por omissão –** Se esta opção tiver activa o nome do ficheiro terá sempre o nome de "lista_classes_geradas_por_treino.txt". Caso contrário deverá ser especificado um nome para o ficheiro de classes desactivando esta opção.
    d. **Processar em simultâneo –** Esta opção diz quantas classes diferentes são processadas em simultâneo. Para que esta opção contenha valores

elevados é essencial o uso de uma quantidade de memória razoável, caso contrário poderá não ser possível processar várias classes em simultâneo. Esta opção é útil para o caso de ficheiros grandes e que requerem grande volume de processamento. Deverá ser usada com critério. Permite processar até um máximo de 5000 classes diferentes em simultâneo.

3. Permir o botão **Criar ficheiro classes** de forma a obter-se o ficheiro de classes.

O ficheiro final a obter será um conjunto de linhas com o seguinte formato:

número classe palavra$_i$ valor,

em que "número" corresponde ao número da classe, "classe" é o nome da classe, "palavra$_i$" é uma das palavras pertencentes a esta classe e "valor" é o valor que essa palavra tem no documento da classe.

A.2 Criação do ficheiro de afinidades

O ficheiro de afinidades pode ser obtido de duas formas diferentes: introduzindo o ficheiro de treino e de teste ou, caso se tenha já processado o ficheiro de classes, introduzindo o ficheiro de classes e o de testes. A primeira forma necessita obrigatoriamente que o ficheiro de classes tenha sido obtido como descrito no ponto anterior.

A.2.1 Ficheiro de classes existente

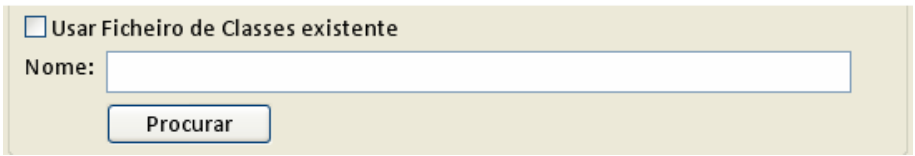

Figura A.3 – Parte da interface utilizada para a introdução do ficheiro de classes.

Para seleccionar o ficheiro de classes deve:

1. Premir-se o botão **Procurar** e escolher o ficheiro de classes gerado no ponto anterior.
2. É necessário activar a opção **Usar Ficheiro de Classes existente**.

A2.2 Introduzir ficheiro de treino

O passo de escolher o ficheiro de treino é o mesmo que no ponto anterior (A.1).

A2.3 Criar o ficheiro de afinidades

Após a execução dos passos anteriores (A2.1 ou A2.2) é necessário agora introduzir o ficheiro de teste. Este é escolhido como exemplificado na Figura A.4.

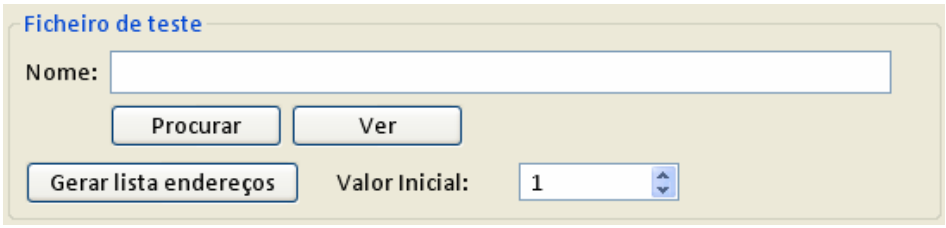

Figura A.4 – Interface para introdução do ficheiro de teste.

Para introduzir o ficheiro de teste deve proceder-se aos seguintes passos:

1. Escolher a opção **Procurar** e escolher o ficheiro de teste.

Finalmente, a criação do ficheiro de afinidades é executada utilizando as opções que estão representadas na Figura A.5.

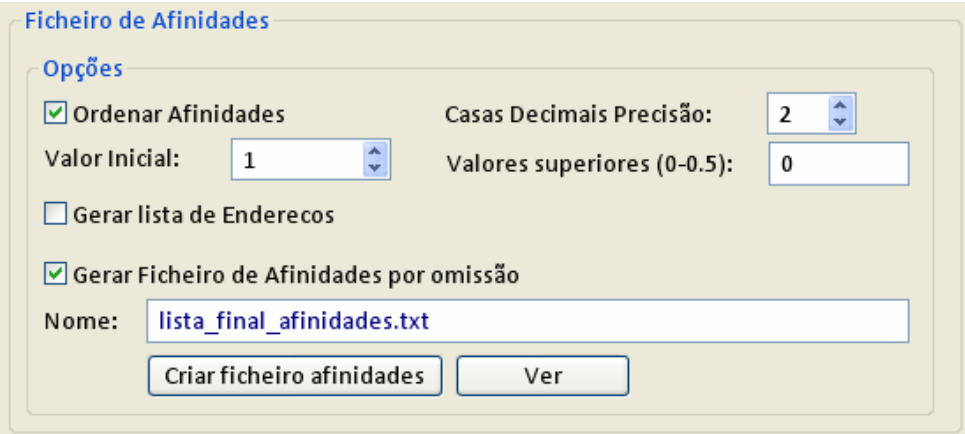

Figura A.5 – Interface para introdução do ficheiro de teste.

As **Opções** disponíveis na criação do ficheiro de afinidades são:

1. **Ordenar afinidades –** esta opção permite, tal como referido na criação do ficheiro de classes, ordenar por ordem decrescente o valor de afinidades.
2. **Casas Decimais Precisão –** O número de casas decimais pretendidas para as afinidades é definida nesta secção.
3. **Valor Inicial** – Esta opção permite escolher a partir de que número o ficheiro de afinidades começará a contagem. Esta opção é importante para o caso de se criar mais documentos de afinidades posteriormente.
4. **Valores superiores (0-0.5) –** Esta opção permite que apenas se contabilizem as classes que tenham afinidade com o documento superior a um determinado valor, sendo o máximo de 0,5. Permite refinar as afinidades para o caso de haver muitos valores de afinidade muito baixos.
5. **Gerar lista de Endereços –** Esta opção permite que, no caso da criação de valores de afinidades para documentos obtidos de endereços da *internet* se gere o ficheiro de Endereços que será usado pelo Babuska para fazer corresponder o documento e o endereço a que este pertence. Esta opção torna mais lento o processo de criação de afinidades.
6. **Gerar Ficheiro de classes por omissão –** Se esta opção tiver activa o nome do ficheiro terá sempre o nome de "lista_afinidades.txt". Caso contrário deverá ser especificado um nome para o ficheiro de classes desactivando esta opção.

O ficheiro gerado terá o seguinte formato:

número ClasseDocTeste ClasseX valorX ClasseY valorY

onde "número" é o número do ficheiro de teste, "ClasseDocTeste" é a classe que originalmente o documento de teste pertence (este nome é só meramente qualitativo), "ClasseX" é uma classe com afinidade com o documento de teste, "valorX" é o valor da afinidade do documento com a classeX, etc.

A.3 Geração do ficheiro de endereços

Uma vez que os ficheiros obtidos da *internet* têm um endereço, este é processao nas diferentes fases do processo e é necessário obter uma relação entre os ficheiros processados e o endereço desse ficheiro para poder ser aberto num motor de busca. Devido à estrutura definida

inicialmente para os ficheiros, este ficheiro faz corresponder o número do documento a um endereço e o título.

Para se obter o ficheiro de endereços existem duas formas; uma descrita na secção A.2, gerando em conjunto com o ficheiro de afinidades, tornado o processo mais demorado, ou, em alternativa, usar a opção descrita na Figura A.4. O ficheiro obtido tem o nome de "Lista_Enderecos.txt" e tem o formato:

número endereço título

A.4 Notas importantes

Este processo pode ser demasiado pesado em termos de utilização de memória, nomeadamente da memória definida pela *virtual machine* do JAVA, sendo por isso necessário aumentar este valor da seguinte forma:

java –Xmx<valor>m –jar <nome_do_programa>.jar

onde:

valor – quantidade de memória em megabytes.

nome_do_programa – nome do jar a executar

Anexo B – Manual para obtenção do conteúdo textual das páginas HTML

Para o processamento de ficheiros contendo código *HTML* ou *JAVA Script*, etc., foi desenvolvido um outro programa que permite limpar o ficheiro de tudo o que não é necessário, ficando apenas o texto disponível para o seu tratamento. A Figura B.1 mostra a interface de interacção inicial com o utilizador.

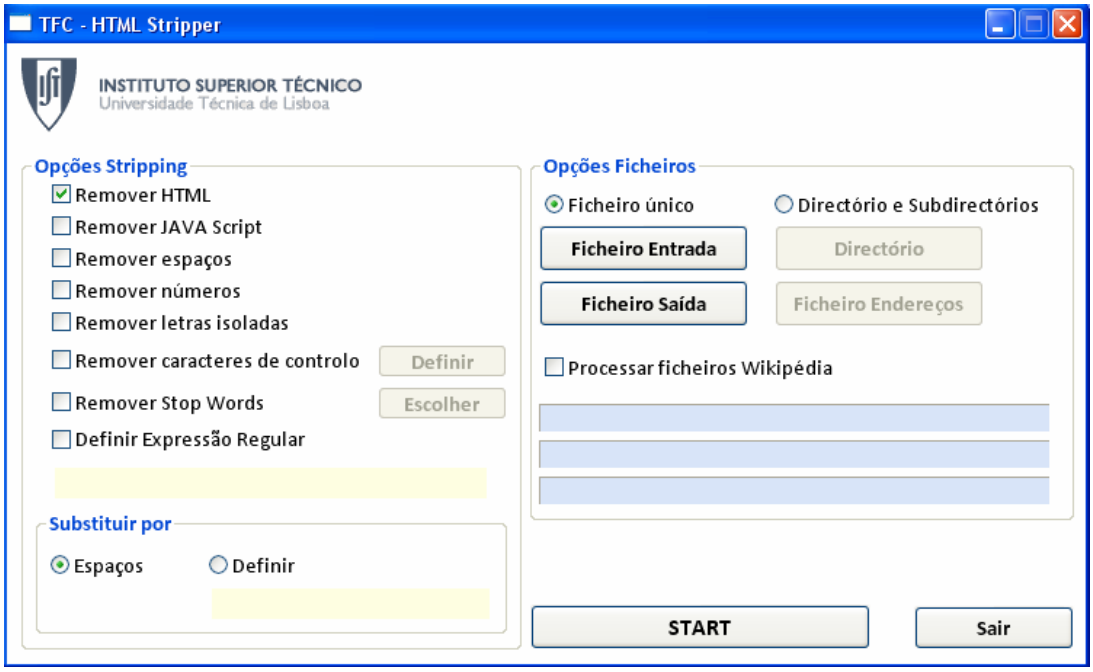

Figura B.1 – Interface principal do programa que permite limpar um ficheiro de elementos *HTML*, *JAVA Script*, entre outros.

B.1 Opções

Estão disponíveis as seguintes opções para remoção de caracteres.

1.1. **Opções de Stripping**

1.1.1. **Remover HTML** –limpa todas as ocorrências de HTML.

1.1.2. **Remover Java Script** –limpa as ocorrências de código JAVA.

1.1.3. **Remover espaços** –limpa espaços extra entre palavras.

1.1.4. **Remover números** – limpa todas os ocorrências de números.

1.1.5. **Ramover letras isoladas** – limpa as ocorrências de letras isoladas .

1.1.6. **Remover Caracteres de Controlo –** Ver B.2

1.1.7. **Remover Stop Words –** Ver B.3

1.1.8. **Definir expressão regular –** O utilizador pode definir uma expressão regular para exprimir qualquer critério não presente nas opções anteriores.

1.2. **Opções Ficheiro**

1.2.1. **Ficheiro Único**

1.2.1.1. **Ficheiro de Entrada –** Ficheiro que vai ser processado. Deve ter extensão .html, .htm.

1.2.1.2. **Ficheiro de Saída –** Ficheiro que irá conter o resultado final do *Stripping* do ficheiro de entrada.

1.2.2. **Directório e Subdirectórios**

1.2.2.1. **Directório –** Directório que contém os ficheiros a ser processados

1.2.2.2. **Ficheiro de Endereços –** Esta opção é necessária para poder ser processado os documentos que provêm de páginas de *internet*.

1.2.2.3. **Ficheiro de Saída –** Ficheiro que irá conter o resultado final do *Stripping* do ficheiro de entrada.

1.2.2.4. **Processar Ficheiros *Wikipédia* –** Esta opção permite usar uma determinada característica dos documentos da *Wikipédia* para melhor o processamento deste ficheiros.

O ficheiro de saída terá diferentes formatos caso seja escolhida, ou não, a opção Ficheiro de Endereços. Esta opção deverá ser activada para se processar os ficheiros de forma a serem usados pelo Babuska. O formato deste ficheiro será:

- Com ficheiro Lista de Endereços escolhido:
  - #html#ENDEREÇO#título#TITULO TEXTO, onde ENDEREÇO é o endereço do documento, TITULO é o título do documento e TEXTO o texto extraído após *stripping*.
- Sem ficheiro Lista de Endereços escolhido:
  - CAMINHO TEXTO, onde CAMINHO é o local onde o ficheiro foi processado e o texto é o texto limpo.

B.2 Controlo

As opções deste módulo são opções que permitem remover caracteres *ASCII* especiais. As opções possíveis neste módulo podem ser visualizadas na Figura B.2.

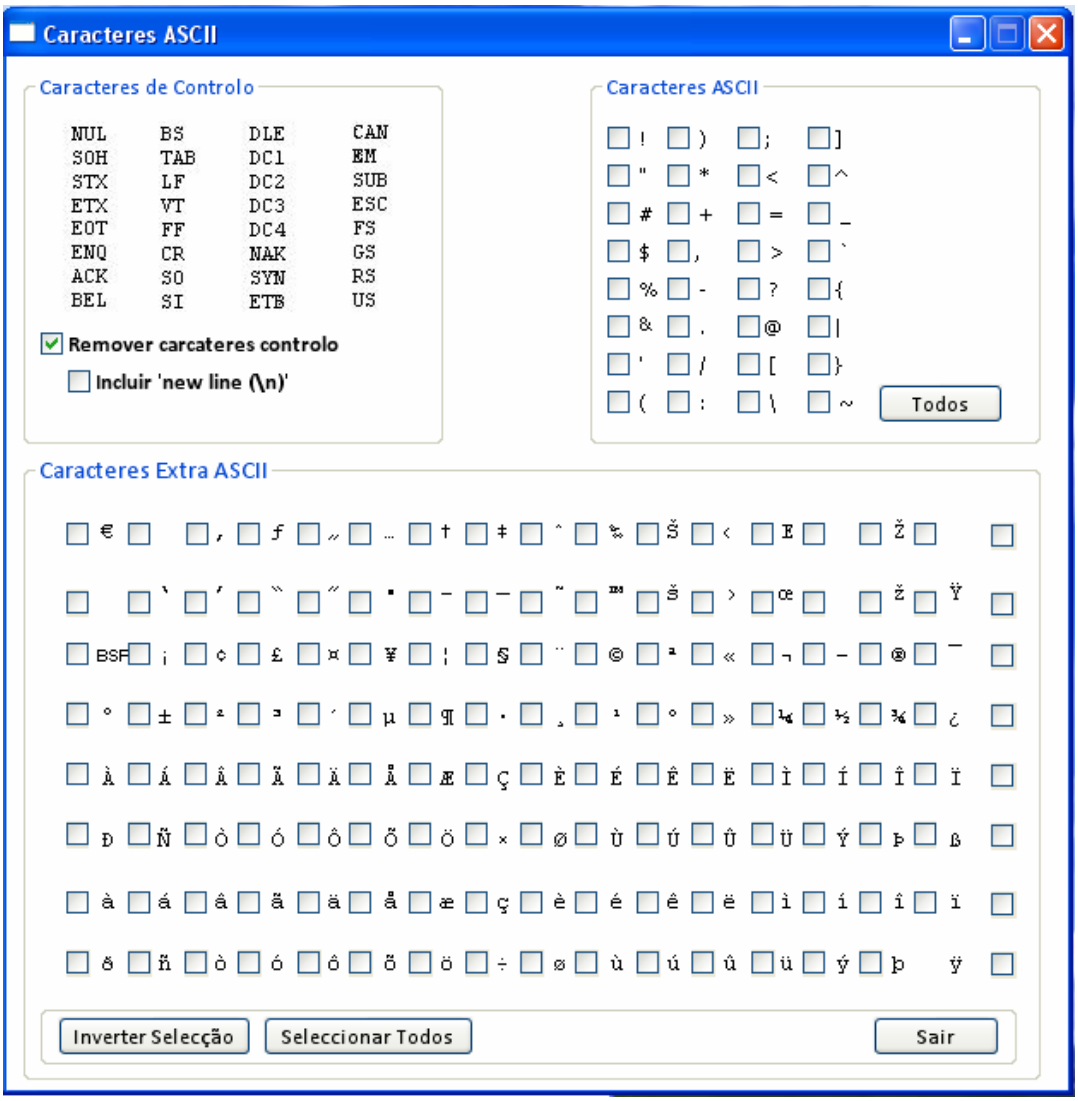

Figura B.2 – Opções disponíveis para remover caracteres especiais.

B.3 Stop Words

Este módulo permite definir *stop words* que devem ser eliminadas do texto a processar. As opções que podem ser seleccionadas estão representadas na Figura B.3.

3.1. ***Stop Words***

3.1.1. As opções disponíveis são Português e Inglês, sendo futuramente possível escolher-se ainda outras linguagens.

3.2. **Carregar *Stop Words*** – Esta opção permite carregar um ficheiro contendo *Stop Words*.

3.3. **Acrescentar *Stop Words*** – Esta opção permite acrescentar outras palavras que não estejam contempladas nas palavras definidas como *stop words*.

3.4. **Excluir *Stop Words*** – Esta opção permite que certas palavras, mesmo que sejam originalmente consideradas como *stop words*, mas que são pretendidas, não sejam assim consideradas para remoção.

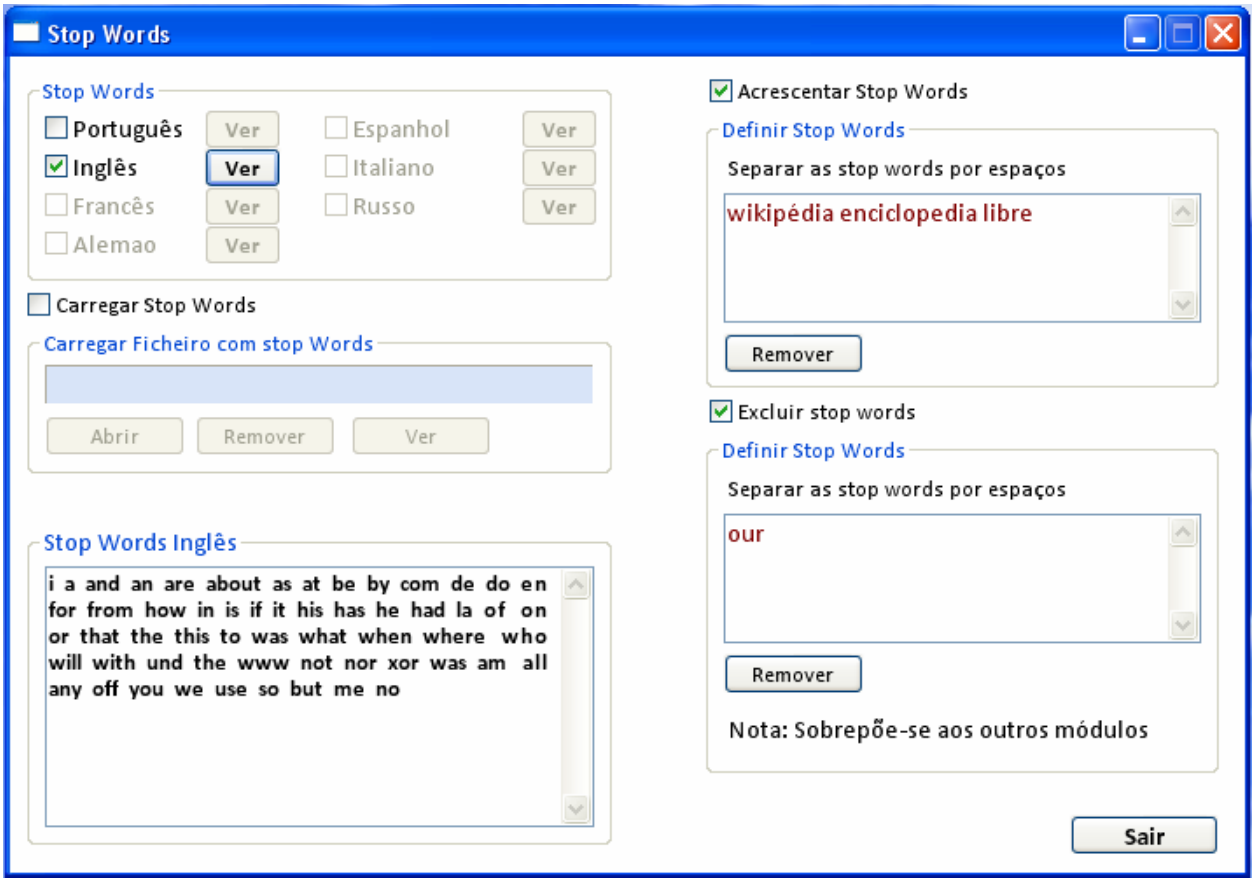

Figura B.3 – Opções disponíveis para remover *stop words*.

Anexo C – Manual para separação de documentos (treino/teste) e indexação

Os seguintes programas permitem escolher a percentagem de documentos processados que serão de treino e quantos serão de teste e permitem também indexar os ficheiros com um dos indexadores disponibilizados pelo *Lucene*.

A Figura C.1 corresponde à interface onde se faz a escolha de um dos programas referidos anteriormente.

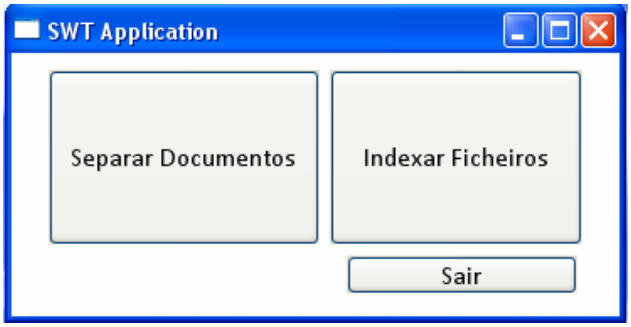

Figura C.1 – Interface para escolha entre os programas Separar Documentos e Indexar Ficheiros.

C.1 Separar Ficheiros

Este programa permite escolher qual a percentagem dos documentos processados (ver Anexo B) vão ser para treino e quantos vão ser para teste. A Figura C.2 mostra a interface deste programa.

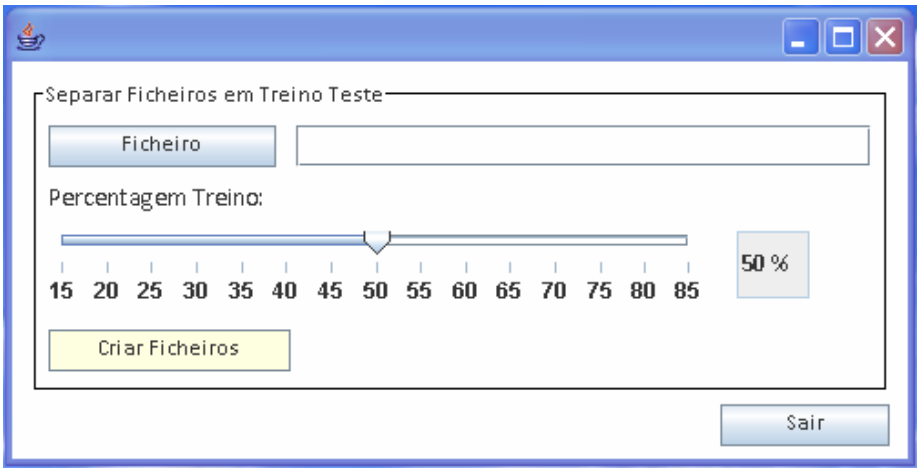

Figura C.2 – Interface para escolha da percentagem dos documentes de treino e de teste.

As opções que podem ser escolhidas neste módulo são:

1. **Ficheiro** – Nome do ficheiro que contém os documentos a processar
2. **Percentagem Treino** – Esta opção permite escolher a quantidade de focumentos que serão usados como treino e quantos serão de teste
3. **Criar Ficheiros** – Após seleccionar esta opção dois ficheiros serão criados. Um será o ficheiro de treino e o outro de teste.

C.2 Indexar Ficheiros

Este programa (Figura C.3) permite que se indexe os ficheiros para o Babuska. Poder-se-á indexar os documentos obtidos do treino e o das afinidades, já devidamente separados em vários ficheiros e indexá-los.

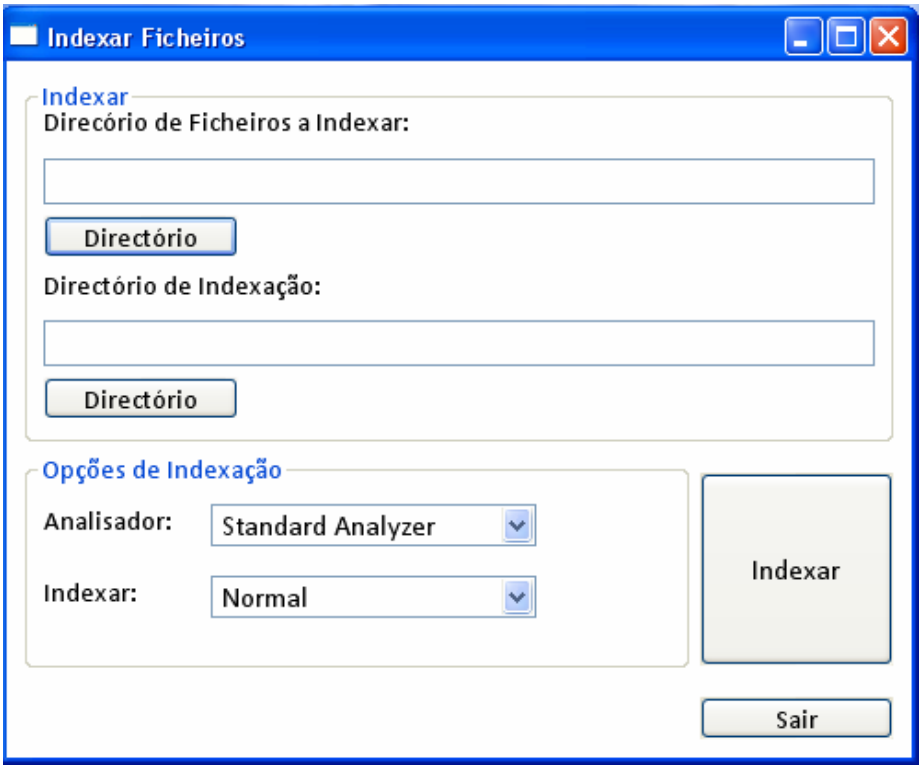

Figura C.3 – Interface para a indexação de ficheiros.

As opções que se podem escolher são:

1. **Directório de Ficheiros a indexar** – Deverá conter o local onde os ficheiros de teste / afinidades foram separados individualmente. Através do botão **Directório** procura-se o directório pretendido.
2. **Directório de indexação** – Corresponde em escolher qual o local para onde se quer indexar os ficheiros. Os ficheiros do Teste para o Babuska directório *index* e os ficheiros das afinidades para o directório *indexAfinidades*.
3. **Opções de Indexação**:
    a. **Analisador** - Permite a escolha entre três analisadores: *Standard*, *Stop* e *WhiteSpace Analyzer*. Estes analisadores estão descritos em [6-7].
    b. **Indexar** – Esta opção permite indexar ficheiros de texto.
4. Botão **Indexar** – Premindo este botão, começa o processo de indexação.

Anexo D – Manual para pré-processamento da Wikipédia

Para se poder processar correctamente a *Wikipédia* e posteriormente poder ser utilizada com o programa de *stripping* (Anexo B) é necessário obter as classes que definem os artigos da *Wikipédia*. Esta necessidade surge porque, tal como se pode ver na Figura D.1, um documento da *Wikipédia* corresponde a várias classes. O ficheiro obtido será então utilizado no programa de *stripping* como Ficheiro de Endereços.

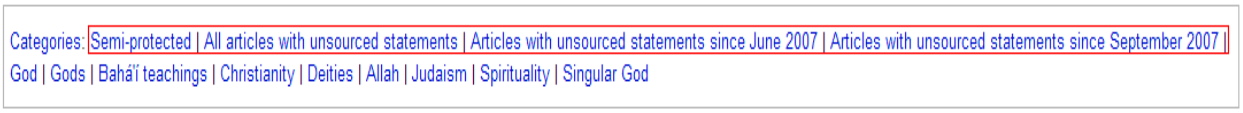

Figura D.1 –Diferentes classes a que um documento pertence. O processamento destas classes é necessário porque existem classificações que não são pretendidas (assinaladas com um rectângulo).

A Figura D.2 corresponde à interface para escolher o directório onde estão os ficheiros da *Wikipédia* obtidos com o *Httrack*. Ao escolher-se **Processar** obtém-se um ficheiro que contém as classes separadas por '@' seguido do endereço.

Figura D.2 - Interface para escolher o directório e começar o processamento dos documentos da *Wikipédia*.

Para o exemplo da Figura D.1 obter-se-ia um documento com as seguintes características:

God@Gods@Bahá'íteachings@Christianity@Deities@Allah@Judaism@Spirituality@SingularGod http://en.*Wikipédia*.org/wiki/God.

Anexo E – Manual para separação de ficheiros

Os ficheiros de teste e afinidades são ficheiros que contêm os diversos documentos que precisam de ser separados de forma a ser possível indexá-los com o *Lucene*. O programa permite que isso seja possível de se fazer. A interface do programa pode ser observada na Figura E.1.

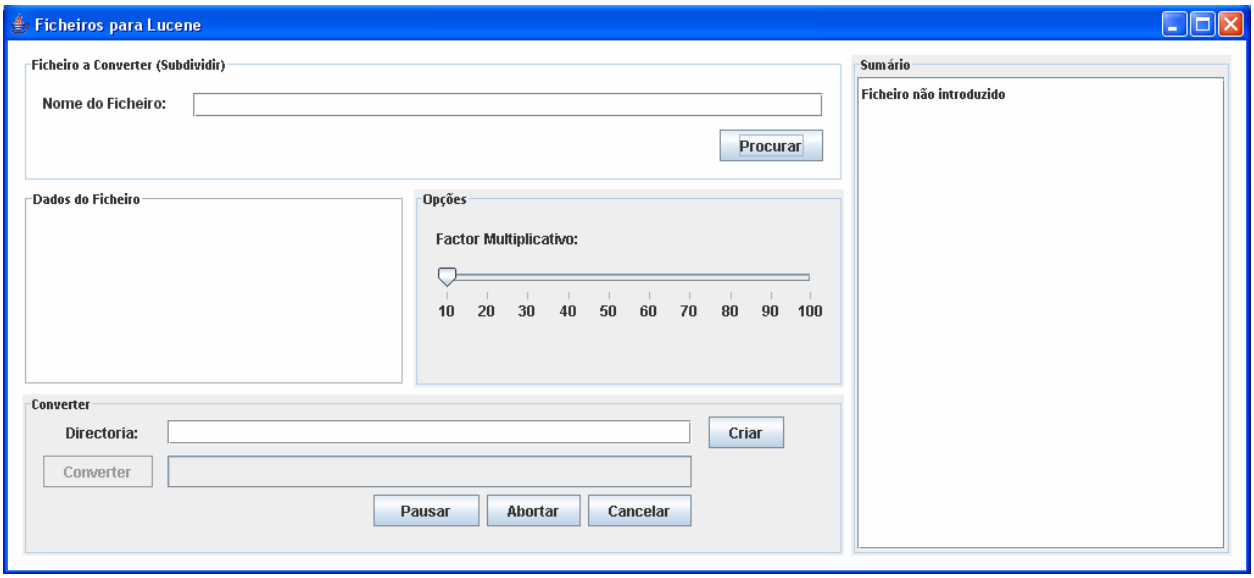

Figura E.1 – Interface do programa que permite converter os ficheiros de teste e de afinidades em ficheiros individuais

Como se pode ver na Figura E.1 existem diferentes secções no programa que a seguir se descrevem.

Os ficheiros a subdividir deverão ser, como já foi referido, os ficheiros de teste e o de afinidades. Estes ficheiros são identificados pelo programa de qual se trata pois têm formatos diferentes.

Deverá seleccionar-se o directório para onde serão escritos os ficheiros. Esta parte é definida na secção **Converter**. Onde se deverá indicar um nome para o directório, introduzido manualmente no respectivo campo.

Uma vez que os ficheiros de afinidades têm um valor de afinidade para cada classe, é necessário indicar qual o valor de multiplicação que se deve escolher para expandir a classe um certo número de vezes. Esta opção é definida em **Opções**. Para ilustrar como funciona esta opção veja-se o exemplo a seguir.

Supondo um documento definido do seguinte modo:

1 casa sala 0,92 cozinha 0,85 quarto 0,63 bolos 0,03 (1)

Ao aplicar o factor multiplicativo de 10, (1) ficará com a seguinte forma:

1 casa sala 9,2 cozinha 8,5 quarto 6,3 bolos 0,3 (2)

Estes novos valores correspondem ao número de vezes que as palavras que definem as classes serão escritas no ficheiro que será o ficheiro "1.txt". Isto é necessário porque o *Lucene* não tem forma de saber que o valor não é para indexar. Assim, (2) fará com que o ficheiro seja escrito da seguinte forma:

sala sala sala sala sala sala sala sala sala cozinha cozinha cozinha cozinha cozinha cozinha cozinha cozinha cozinha cozinha quarto quarto quarto quarto quarto quarto quarto.

Anexo F – Manual para ver e exportar dados obtidos por Avaliação no Babuska

Os resultados obtidos após avaliação têm de ser processados de forma a extraír-se informação útil dos mesmos. Para que se consiga perceber se uma pesquisa é melhor que outra é necessário avaliar os resultados, como foi visto no capítulo 5. Para que estes dados sejam tratados é necessário condensar os resultados por pesquisas: Textual, Semântica e Combinada.

O programa seguinte permite que se processe o ficheiro de resultados e ainda permite criar um ficheiro que pode ser lido por um programa gráfico, ou que permita análise gráfica, e assim os dados recolhidos podem ser usados de forma analítica.

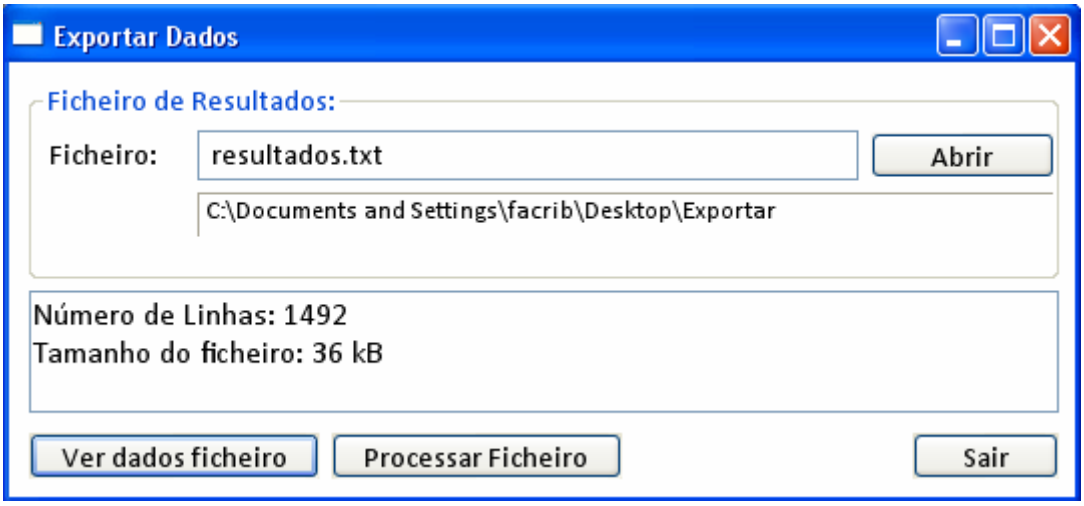

Figura F.1 – Interface gráfica de interacção com o utilizador.

A Figura F.1 corresponde à interface gráfica com que o utilizador interage inicialmente. É aqui que será aberto o ficheiro de resultados que será posteriormente processado. A figura mostra um ficheiro (resultados.txt) que foi aberto e o respectivo caminho para esse ficheiro (C:\Documents....). Os comandos que se podem executar nesta interface são:

1. **Abrir** – Abre um ficheiro para posterior processamento. Este ficheiro tem obrigatoriamente de ser o ficheiro de resultados obtidos através do motor de busca Babuska e da avaliação feita às pesquisas efectuadas.

2. **Ver dados ficheiro** – Permite ver o tamanho e o número de linhas a processar do ficheiro. Caso não tenha sido escolhido nenhum ficheiro uma mensagem de erro aparecerá no ecrã.
3. **Processar Ficheiro** – Ao seleccionar-se esta opção outro ecrã aparecerá com os resultados processados e que pode ser visto na Figura F.2.
4. **Sair** – Sai do programa.

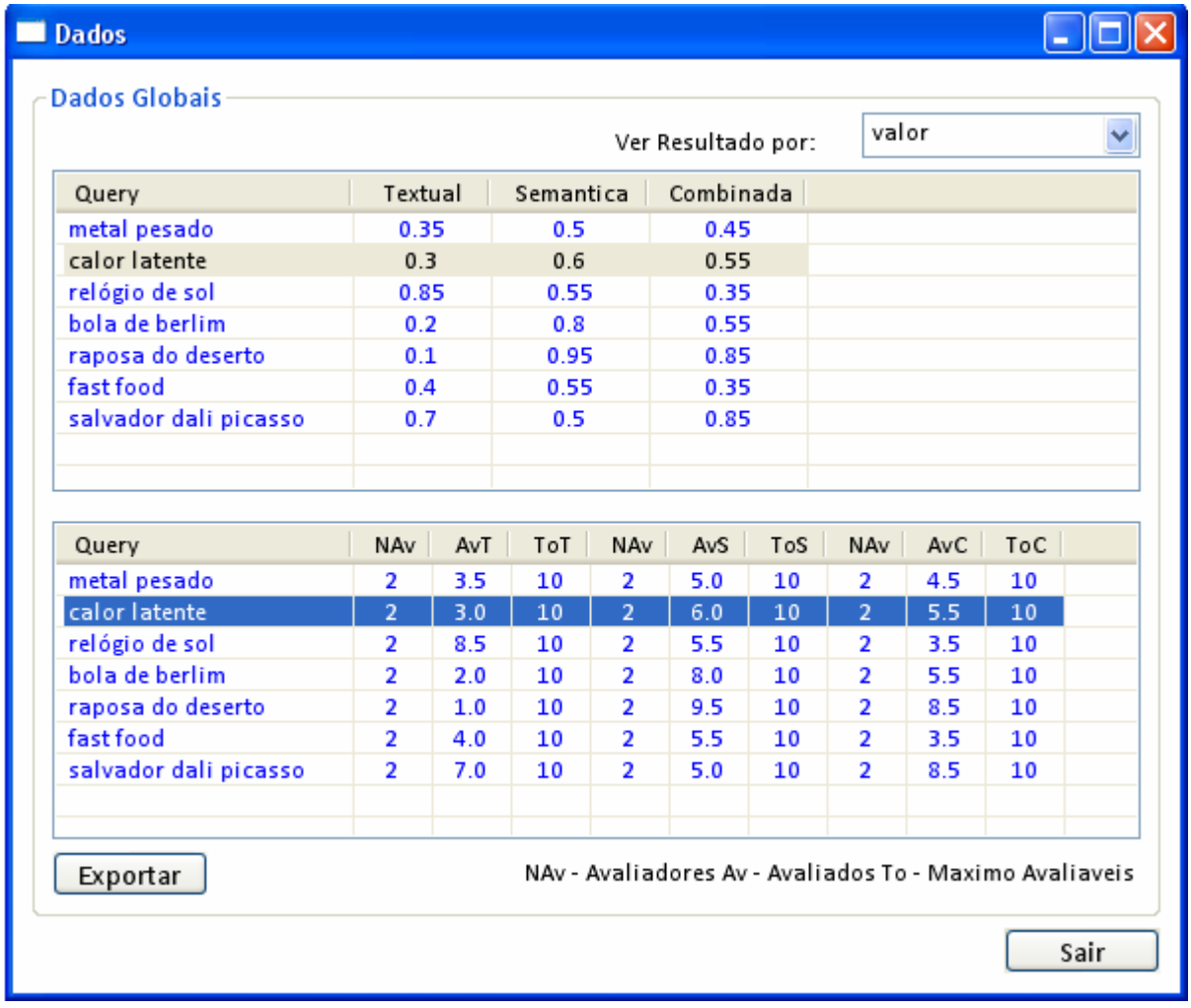

Figura F.2 – Interface onde se apresentam os dados processados do ficheiro.

A Figura F.2 apresenta os resultados processados. O quadro superior mostra os valores médios das boas classificações atribuídas pelos avaliadores nas várias expressões de busca na pesquisa textual, semântica e combinada, ao passo que a tabela inferior mostra o número de

avaliadores que avaliaram as expressões de busca, a médias dos resultados "bons" obtidos e o número máximo de resultados que poderiam ser avaliados para cada um dos tipos de busca (este número depende dos resultados que o *Lucene* consegue pesquisar).

Para facilitar uma leitura dos quadros mais fácil é possível seleccionar uma das linhas de um dos quadros e ver qual a correspondente no outro quadro (linhas preenchidas com fundo na Figura F.2).

As possíveis interacções nesta interface são:

1. **Exportar** – Cria um ficheiro com extensão .xls com os dados das tabelas e que pode ser aberto em programas gráficos para futuro tratamento analítico.
2. **Ver Resultado por** – Alterna a representação no quadro superior por valores (precisão) e percentagem dos resultados.
3. **Sair** – Sai do programa.

Anexo G – Manual para classificador de documentos

O classificador é utilizado para verificar se as expressões de busca transformadas dão origem a documentos com afinidades mais próximas das esperadas ou se dão origem a afinidades pouco ou nada relacionadas com a expressão de busca.

O programa começa com uma interface onde se escolhe qual o classificador (ficheiro de classes obtidos a partir dos documentos de treino tal como descrito na secção 4.1.2) a usar. O calssificador a usar é um dos definidos na *drop down box* e é a opção **Classificadores**. Esta interface pode ser observada na Figura G.1.

O botão **Refresh** permite verificar se o directório onde estão os classificadores contém novos classificadores, e introduzindo os novos classificadores na *drop down box*.

O botão **Classificar texto** conduz a um outro ecrã onde será possível introduzir expressões de busca e obter a afinidade desta com as classes que estão definidas no classificador. A interface para o classificador está representada na Figura G.2.

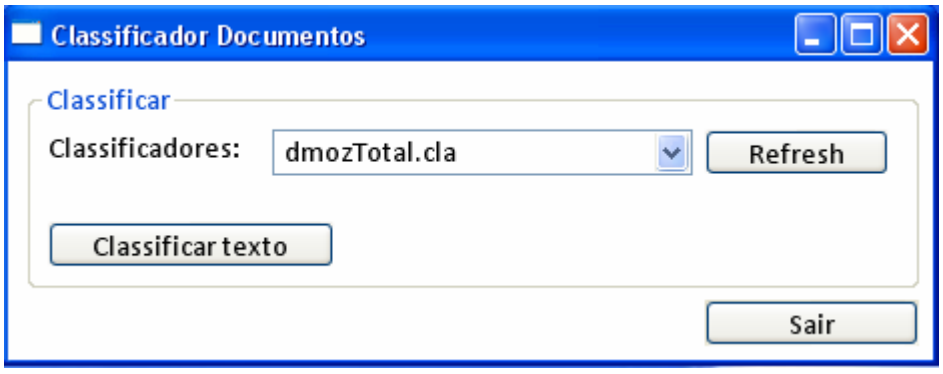

Figura G.1 – Interface para escolha do classificador a utilizar.

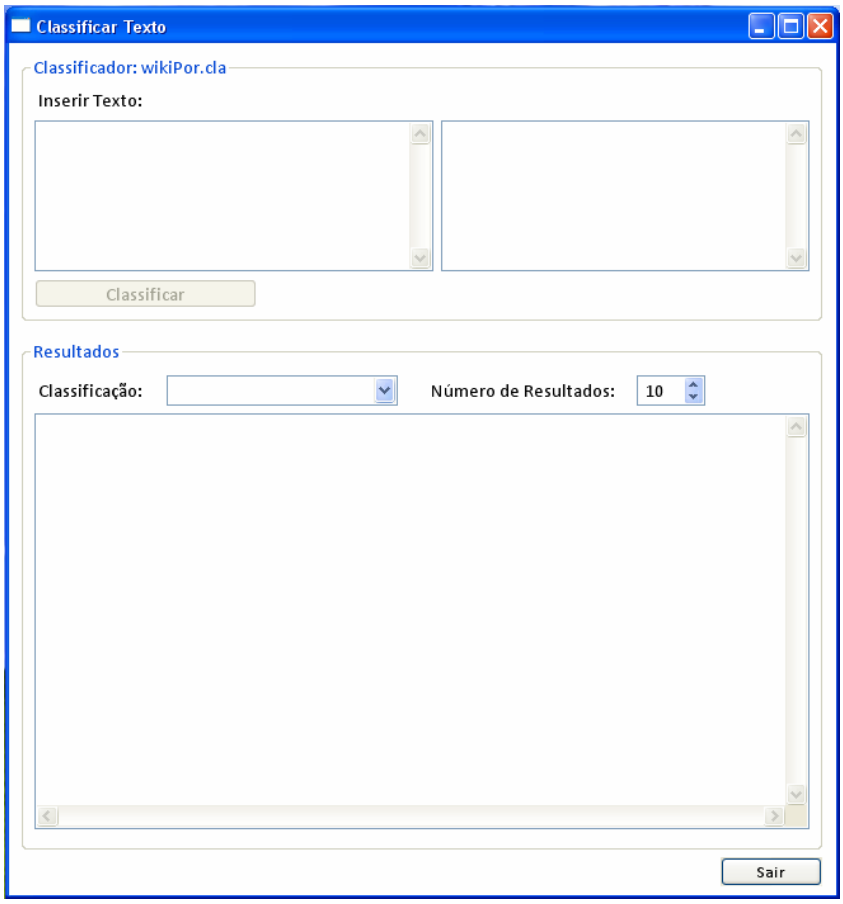

Figura G.2 – Interface onde é possível introduzir uma expressão de busca e obter a afinidade com as classes do classificador.

A Figura G.2 corresponde ao local onde a interacção é feita de forma a obter-se as afinidades. Esta interface está dividida em 2 secções.

A primeira secção é a que corresponde à introdução da expressão de busca. O título corresponde ao Classificador em uso, que no caso da Figura G.2 é o da *Wikipédia* portuguesa. No campo **Inserir Texto** deve escrever-se a expressão de busca que se pretende classificar com vista a obter as classes com as quais tem afinidade. No quadro adjacente ficará um registo das acções.

Para se classificar o texto introduzido deve premir o botão **Classificar**. Este botão apenas fica disponível em caso da introdução de texto na janela **Inserir texto**.

A segunda secção é onde os resultados serão apresentados. Esta secção corresponde à caixa **Resultados** e apresenta algumas opções:

1. **Classificação** – Esta *drop down box* vai conter as opções de classificação obtidas e depende do número de palavras que compõem a expressão de busca. Pode ter até um máximo de 5 tipos de opções, variando entre Melhores Resultados e Outros Resultados.
2. **Número de Resultados** – Esta opção permite seleccionar quantos resultados serão apresentados em simultâneo, num máximo de 30 resultados para o caso destes existirem.

A janela de texto na secção **Resultados** mostra os resultados obtidos.

G.1 Como é feita a classificação

Após a introdução da expressão de busca os resultados obtidos dependem do número de palavras da expressão de busca. Primeiramente é verificado quantas palavras serão utilizadas no processamento e quantas é que têm maior correspondência nas classes tal como descrito na secção 4.4.3. Os resultados obtidos são função deste número. Um exemplo:

Classificar a expressão de busca "bola de berlim". Na Figura G.3 podemos ver a para esta expressão de busca existem 2 palavras a processar pois o "de" é uma *stop word* e é eliminada. Como não há palavras repetidas o número de palavras a processar é o mesmo da expressão de busca e o número máximo de correspondência também é de 2 palavras.

Figura G.3 – Expressão de busca "bola de berlim" e o respectivo registo.

Os resultados obtidos podem ser observados na Figura G.4, onde se vê a correspondência com as duas palavras e que corresponde aos resultados mais fiáveis. Note-se a quantidade de classes em que o documento pode ser classificado. A Figura G.5 mostra resultados aproximados, onde apenas uma correspondência foi observada, isto é, ou uma ou outra palavra é que pertencia à classe.

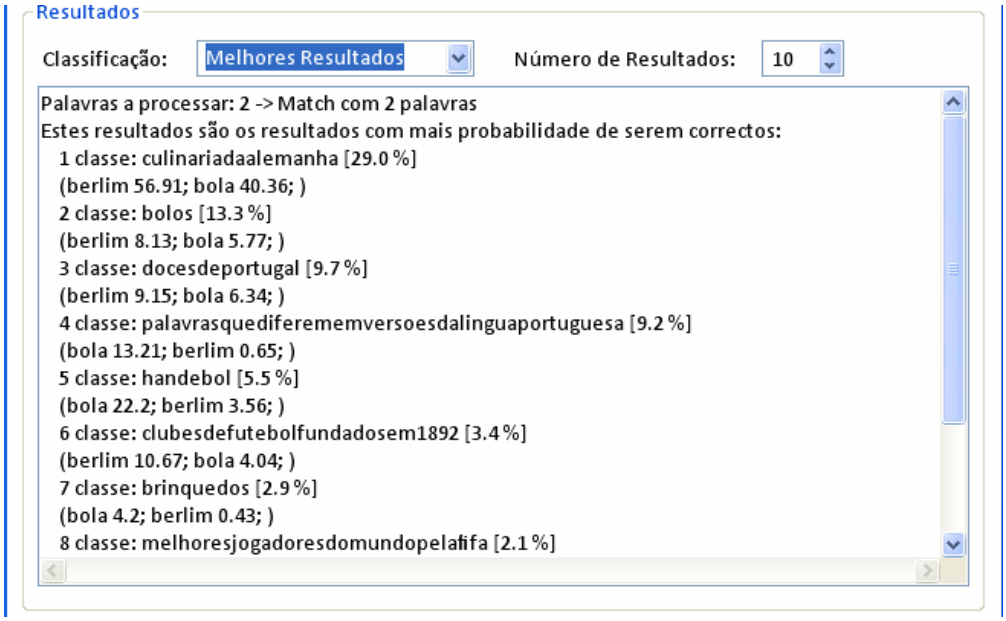

Figura G.4 – Classificações obtidas com uma correspondência total para "bola de berlim".

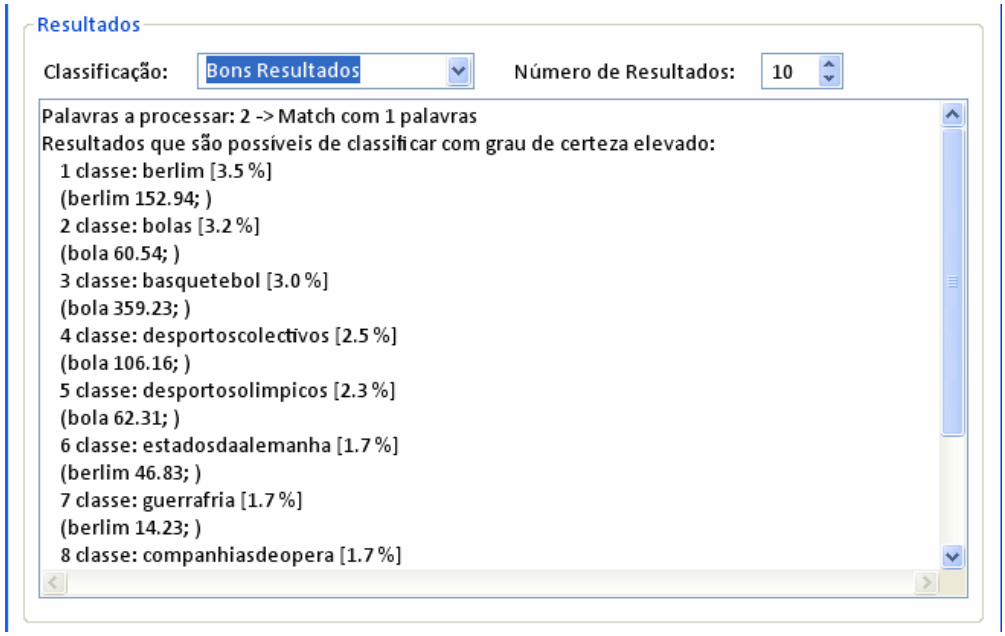

Figura G.5 – Classificações obtidas com uma correspondência de uma palavra para "bola de berlim".

Anexo H – Procedimento para criação das colecções utilizadas

Para a obtenção dos ficheiros necessários à realização deste trabalho, teve de haver um processamento de todas as colecções utilizadas. Devido ao formato específico de cada colecção foi necessário um processo de uniformização inicial. Os procedimentos para cada uma das diferentes colecções encontram-se descritos de seguida.

H.1 Colecções *Reuters* e *Cadê*

Para estas colecções foram seguidos os seguintes passos.

1. Processar os ficheiros de treino e teste como descrito no Anexo A.

   Obtêm-se dois ficheiros, o de classes e o de afinidades (ver secção 4.1.2)

2. Usar o programa descrito no Anexo E para separar o documento de teste e o documento de afinidades gerado no ponto anterior para criar ficheiros individuais (O *Lucene* indexa ficheiros individualmente).

   Criam-se duas directorias para onde serão escritos os documentos como ficheiros individuais. Uma será para os ficheiros de teste e as outras para os ficheiros de afinidades.

3. Indexar os ficheiros de teste e de afinidades, utilizando o programa descrito no Anexo C para indexar, devendo ser indexado os ficheiros separados do teste para o directório *index* e os ficheiros de afinidades separados para o *indexAfinidades* do directório onde o Babuska está.

4. Colocar no directório **ficheiros** do Babuska o ficheiro de classes com o nome "lista_classes.txt".

H.2 Colecção *Wikipédia*

Estes ficheiros foram processados da seguinte forma:

1. Processar os ficheiros da *Wikipédia*, segundo o procedimento descrito no Anexo D. Obter um ficheiro com endereços e classes

2. Processar os ficheiros da *Wikipédia* com o programa de *stripping* segundo o Anexo B, utilizando o programa obtido no ponto anterior como Ficheiro de Endereços (ver secção 4.1.2) e seleccionando a opção de processar como directórios.
   Obter o ficheiro de com as classes, endereço, título e texto limpos de *HTML*, *JAVA Script*, etc.
3. Separar os ficheiros obtidos no ponto anterior em ficheiros de treino e teste. Por motivos de espaço, memória e tempo de processamento, o valor optado para o treino foi de 30 % e para o teste de 70 %. Esta separação é efectuada segundo o procedimento descrito no Anexo C (secção Separar Documentos).
   a. Obtenção de dois ficheiros. Um de treino e outro de teste.
4. O procedimento a seguir é de apagar as linhas duplicadas no ficheiro de teste. Não existe um programa definido para o efeito podendo para tal usar-se um programa que permita usar expressões regulares que façam o pretendido. Para tal todas as ocorrências com o mesmo endereço e o mesmo título devem ser eliminadas do conjunto de teste.
   Este procedimento é necessário para o teste porque aqui não estão em causa as classes todas, o documento irá ter afinidade com diversas classes e isso é definido segundo o ficheiro de treino.
5. Processar o ficheiro de treino obtido no ponto 3 e o de teste no ponto 4 para obter os ficheiros de classes e afinidades, aplicando um dos dois métodos descritos no programa descrito no Anexo A. Recomenda-se no entanto a criação do ficheiro de classes e só depois originar o ficheiro de afinidades para evitar tempo de processamento desnecessário.
6. Após a criação dos ficheiros de classe e de afinidades no ponto anterior é necessário separar os documentos de cada um destes ficheiros em ficheiros separados e isso é feito segundo o programa descrito no Anexo E.
7. Indexar os ficheiros separados utilizando para tal o procedimento descrito no Anexo C (Indexar Ficheiros).
8. Colocar os respectivos ficheiros no directório do Babuska. Os ficheiros de classes e endereços no directório ficheiros.

H.3 Colecção *Dmoz*

Para se processar os ficheiros da colecção do *Dmoz* deve seguir-se o seguinte procedimento.

1. Fazer a descarga do ficheiro "rdf" em [13]. Obter um ficheiro com os endereços que se querem processar no Httrack.
2. Utilizar o ficheiro obtido em 1 no programa *Httrack* e obter as páginas com os urls especificados.
3. Processar os ficheiros do *Dmoz* com o programa definido em Anexo B, utilizando como ficheiro de endereços o ficheiro obtido em 2, terminando em _Out.txt.
4. Separar o ficheiro obtido em 3 em ficheiro de treino e ficheiro de teste. Para tal, usar o programa definido no Anexo C, escolhendo um valor de percentagem para o treino e teste.
5. Processar os ficheiros obtidos no ponto 4 com o programa obtido no Anexo A. Obtém-se o ficheiro de classes e o de afinidades. Obter também o ficheiro da lista de Endereços.
6. Separar os ficheiros de teste e afinidades tal como descrito no Anexo E.
7. Indexar os ficheiros obtidos em 6 com o programa descrito no Anexo C (Indexar ficheiros).
8. Colocar os ficheiros de classes e afinidades no directório do Babuska, mais precisamente no directório ficheiros.