Pesquisador Multi-Agente na Web

André Barbosa
Faculdade de Engenharia
Universidade do Porto
Porto, Portugal
+351 91 823 50 68
ei01043@fe.up.pt

Filipe Montenegro
Faculdade de Engenharia
Universidade do Porto
Porto, Portugal
+351 91 630 42 14
ei01104@fe.up.pt

SUMÁRIO
Inserido no âmbito da disciplina de Agentes e Inteligência Artificial Distribuída do 4º ano da Licenciatura em Engenharia Informática e Computação da Faculdade de Engenharia da Universidade do Porto o Pesquisador Multi-Agente na *Web* retende introduzir o conceito de inteligência artificial presente em diversos corpos (agentes), cada um deles com a capacidade de tomar as suas próprias decisões e de executar diferentes acções consoante o pretendido. O projecto visa uma execução de pesquisas específicas na *web*, isto é, dentro de todas as pesquisas possíveis para uma determinada palavra a aplicação tem de ser capaz de dividir os contextos a que essa palavra se refere fazendo depois uma sub-pesquisa dentro dessa área. Importa referir que cada vez mais este assunto tem sido tema de estudo por diversas pessoas o que verifica a sua importância na sociedade e no contexto actual.

1. OBJECTIVO
O sistema multi-agente é constituído por diversos agentes cuja função é pesquisar páginas web de acordo com palavras-chave fornecidas pelo utilizador e do perfil do próprio utilizador (conjunto de preferências ou palavras-chave armazenadas). A pesquisa de informação é efectuada a partir de um endereço inicial, definido pelo utilizador, ou através dos endereços de *n* páginas retornadas pelo google. O agente pesquisador deve analisar não só a(s) página(s) inicial(is), mas também os seus "links" para verificar se as novas páginas contêm as palavras a procurar. Deve existir um limite máximo de profundidade das "sub" páginas a pesquisar.

Deve ser dada ao utilizador a possibilidade de classificar algumas páginas encontradas para guiar o pesquisador na sua tarefa. Essa classificação pode ser um de dois valores: interessa/não_interessa. O algoritmo de análise pode conter sub-algoritmos que considere relevantes para melhor qualificar as páginas encontradas, tais como: realizar a eliminação de palavras desnecessárias e obter palavras derivadas. A qualificação de uma página deriva da consideração de número de palavras-chave encontradas, posição das palavras na página, e do perfil do utilizador.

2. DESCRIÇÃO
Todos os aspectos importantes quer a nível da execução como da implementação do sistema são apresentados neste capítulo. Criamos diversos diagramas de sequência, de classes e *UML* para uma melhor compreensão das diversas componentes da aplicação e estruturamos o capítulo em 5 partes diferentes: funcionalidades, estrutura do programa, esquemas de representação conhecimento, detalhes da implementação e ambiente de desenvolvimento. Todas estas componentes estão descritas nos tópicos seguintes pormenorizadamente.

2.1 Funcionalidades
O menu principal da aplicação, **Search Text** apresenta ao utilizador todas as opções com que a pesquisa e os resultados finais podem ser definidos. Após a submissão deste diversos menus ficam disponíveis para apresentação de resultados tais como **Initial Results Table**, **Agents Table**, **Words Table** e **Words Cluster Selection**. No último menu o utilizador procede à selecção de um contexto específico e a aplicação procede a uma nova busca apresentado os resultados finais em Final Results. Todos os menus referidos neste parágrafo estão descritos nos tópicos seguintes:

2.1.1 Search Text
Este é o menu principal do programa e nele são definidas todas as características da pesquisa bem como todos os dados necessários para a execução do algoritmo de agrupamento em **clusters**. Os campos deste menu encontram-se definidos a seguir:

- **Search Text** – texto a pesquisar
- **Initial Results** – nº de resultados provenientes do Google (1-10)
- **Depth Level** – nível de profundidade dos agentes (1-5)
- **Words Orientation** – orientação da pesquisa de palavras (Full Page ou Sentences)
- **Algorithm** – tipo de algoritmo para o agrupamento em clusters (1 ou 2)
- **Words** – nº de palavras a ser usadas para o agrupamento em clusters (10, 20, 50, 100, 200 ou 500)
- **Delta** – valor do intervalo usado no algoritmo para agrupamento em clusters (0.01, 0.03, 0.05, 0.075, 0.1, 0.3, 0.5 ou 0.75)
- **Search** – define o alcance da pesquisa (the web ou pages from Portugal)

Após pressionar o botão **Submit** a pesquisa é iniciada e todos os menus dos resultados intermédios retornados.

2.1.2 Initial Results Table
Neste menu intermédio são apresentados todos os resultados provenientes do *Google* que serão utilizados pelos agentes de pesquisa como endereços iniciais. A visualização dos resultados é

feita através de uma tabela onde a 1ª coluna (*URL*) diz respeito ao endereço e a 2ª (*State*) indica se este está disponível para pesquisa ou não (*Enabled* / *Disabled*), mostrando assim se o endereço será utilizado por um agente de pesquisa. Além da tabela com os resultados do *Google* é também apresentado o tempo que este motor de busca demorou a efectuar a pesquisa no campo **Google Search Time**.

2.1.3 Agents Table

Todos os agentes de pesquisa lançados pela aplicação são listados numa tabela nesta secção que é composta pelas colunas **Name**, **Initial URL**, **Words** e **Links**. A coluna **Name** diz respeito ao identificador que o agente possui no sistema, o endereço inicial de pesquisa é apresentado na coluna **Initial URL** e o nº de palavras e endereços encontrados durante a pesquisa por cada agente são indicados nas colunas **Words** e **Links**. O tempo que os agentes demoraram a efectuar a pesquisa encontra-se no campo **Agents Time.**

2.1.4 Words Table

Componente da aplicação que recorre à base de dados para mostrar todas as palavras encontradas na pesquisa bem como o nº de ocorrências de cada uma. Assim a tabela é constituída pelas colunas **Word** e **Ocurrances** que indicam os valores descritos anteriormente. Tal como em todos os menus intermédios anteriores também este disponibiliza o tempo de processamento **Words Time**, que neste caso corresponde ao tempo que as palavras demoraram a ser actualizadas na base de dados.

2.1.5 Words Cluster Selection

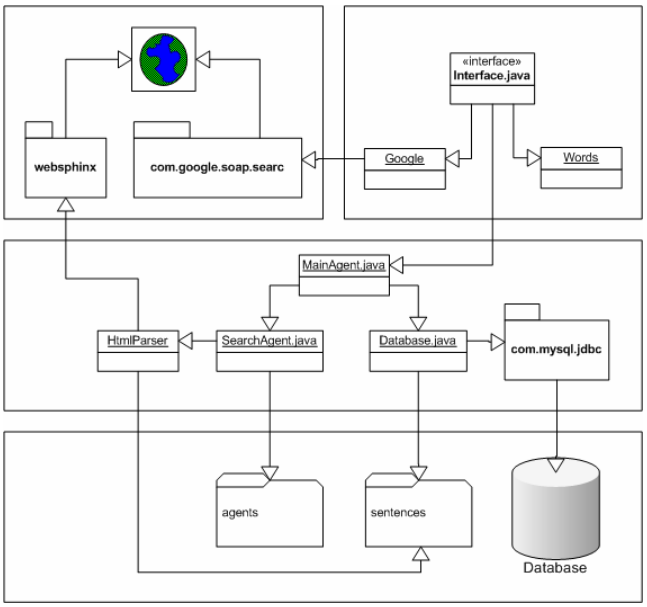
Após a aplicação do algoritmo de agrupamento em **clusters** uma lista com todos os **clusters** encontrados aparece neste menu intermédio para posterior selecção por parte do utilizador. A tabela correspondente a esta parte do sistema é composta pela coluna **Name**, que indica o nome do **cluster** e pela coluna **Words** que contém uma listagem de todas as palavras presentes no respectivo **cluster**. O utilizador selecciona o **cluster** pressionado o rato na linha correspondente ao **cluster** pretendido e uma mensagem de confirmação surge perguntando se realmente pretende fazer uma nova pesquisa com os novos dados. Se a resposta for afirmativa a aplicação realiza uma nova pesquisa e apresenta os resultados finais.

2.1.6 Final Results

Este é o menu final do sistema que, após a selecção de um determinado contexto (**cluster**), apresenta ao utilizador os resultados finais, ou seja, todos os sítios web que contém as palavras existentes no **cluster** seleccionado com a restrição de apenas procurar 10 palavras do contexto, limite implementado em todas as pesquisas no *Google*. Assim, a tabela correspondente aos resultados finais é constituída pelas colunas *URL* e *State* que indicam o sítio web respectivo e o estado da página (*Enabled* / *Disabled*). O tempo de pesquisa do Google é indicado no campo **Google Search Time**, tal como nos resultados iniciais.

2.2 Estrutura do Programa

Relativamente à estrutura do programa podemos dizer que, de uma maneira geral, este se encontra divido em 4 camadas: camada web, camada de controlo, camada de pesquisa e camada da base de dados. Para uma melhor compreensão da nossa divisão apresentamos a seguir um diagrama *UML* bem como a descrição de cada uma das camadas.



2.2.1 Camada Web

Esta camada diz respeito à comunicação que o nosso sistema realiza com a web sendo esta composta por 2 pacotes externos ao nosso sistema: **websphinx** e **com.google.soap.search**. O primeiro é relativo à identificação de links e palavras numa página web e o segundo diz respeito à *API* do *Google*. Assim, apesar de esta camada não ter sido implementado por nós, é importante referir a sua presença visto ser vital na execução do programa.

2.2.2 Camada de Controlo

A interface, o lançamento dos agentes e a aplicação do algoritmo de agrupamento em **clusters** é feita nesta componente que demos o nome de camada de controlo visto ser ela que procede, de uma maneira, ao controlo do sistema. Assim, esta é composta pela interface com o utilizador, pela interface com o *Google* e pela classe respectiva ao algoritmo de contextos. O lançamento dos agentes é feito em background através da execução de uma nova **shell** que, por sua vez, chama o jade com um agente inicial.

2.2.3 Camada de Pesquisa

A análise dos endereços e das palavras contidas numa página web é feita nesta camada da aplicação pelos agentes de pesquisa que são lançados pelo agente principal. Esta recolha de informação é feita em conjunto por um **parser** desenvolvido por nós e por um pacote externo desenvolvido para efeitos semelhantes. Toda a informação recolhida é armazenada na base de dados e nos ficheiros correspondentes utilizando para tal uma classe que faz a interface com a camada da base de dados e um pacote externo que permite a utilização de bases de dados **mysql**.

2.2.4 Camada da Base de Dados

Esta é a camada mais baixa do sistema e representa a base de dados **mysql** presente no sistema e todos os ficheiros utilizados para o armazenamento de frases lidas das páginas *web*. Por sua vez todos os agentes da aplicação mantêm um histórico de

execução que permite analisar posteriormente todas as suas acções e percepções.

2.3 Esquemas Representação Conhecimento

A base da aplicação, o algoritmo de agrupamento em clusters, será descrita neste tópico bem como todos os dados relevantes para a obtenção de conhecimento. Assim, consideramos que esta componente da aplicação se divide em 2 fases: a construção da matriz de ocorrências e o agrupamento em **clusters**. Estas fases são descritas abaixo.

2.3.1 Matriz de Ocorrências

A primeira fase do programa para o agrupamento em **clusters** das palavras é a construção da matriz de co-ocorrências. Esta construção é feita através dos seguintes passos:

- Inicialmente é criada uma matriz com tamanho igual ao nº de palavras seleccionadas pelo utilizador para agrupar em **clusters** com valores iniciais em todas as posições igual a zero.
- No segundo passo a aplicação procede à recolha das N (tamanho da matriz) palavras com maior nº de ocorrências para um lista mantida em memória.
- Depois a matriz é percorrida e, para cada par de palavras, todas as frases existentes são analisadas e o sistema verifica se ambas as palavras ocorrem na mesma frase. Se ocorrerem o valor dessa posição é incrementado senão simplesmente ignora.

2.3.2 Agrupamento em Clusters

O algoritmo utilizado no nosso programa diz que para uma palavra B pertencer a um **cluster** que contém a palavra A o valor da fórmula nº de ocorrências entre A e B / (nº de ocorrências de A + nº de ocorrências de B – nº de ocorrências entre A e B) tem de ser menor que o valor delta (definido pelo utilizador). Assim, a aplicação começa inicialmente com uma lista vazia onde serão adicionadas lista em cada posição correspondentes aos diferentes **clusters**. Para cada palavra que ainda não possui **cluster** consoante o valor retornado da fórmula e posterior verificação está é adicionada ou não a um **cluster** podendo diversas palavras ficarem isoladas num **cluster**.

2.4 Detalhes de Implementação

Devido aos problemas que tivemos com a grande quantidade de dados tratados e com o tempo excessivo de processamento optámos por armazenar grande parte da informação numa base de dados **mysql** e em ficheiros de texto para libertar memória. Assim, apenas algumas listas são mantidas durante a execução do programa para obter um acesso mais rápido. De todas as estruturas utilizadas no desenvolvimento do sistema importa salientar as tabelas da base de dados, os ficheiros que armazenam as frases e os ficheiros que mantêm os históricos dos agentes. Estas estruturas estão descritas nos tópicos a seguir apresentados.

2.4.1 Tabelas da Base de Dados

AIAD_LINKS	
PK	code
	link
	priority

AIAD_AGENTS	
PK	code
	name
	initial
	words
	links

AIAD_WORDS_COPY	
PK	code
	word
	ocurrances

AIAD_WORDS	
PK	code
	word
	ocurrances

A tabela **AIAD_LINKS** guarda todos os endereços analisados pela aplicação e possui os campos **code** (identificadore do endereço), **link** (endereço) e **priority** (prioridade do endereço). Os agentes de pesquisa lançados são mantidos na tabela **AIAG_AGENTS** que além do código identificador **code** possui também um campo **name** (nome do sistema), **initial** (endereço inicial de pesquisa), **words** (nº de palavras encontradas pelo agente) e **links** (nº de endereços encontrados pelo agente). A utilização da tabela **AIAD_WORDS_COPY** surgiu apenas na sequência de diversos problemas que ocorreram devido ao facto de estarmos a fazer acessos simultâneos à base de dados, o que impossibilitava o controlo de palavras repetidas. Assim, nesta tabela todas as palavras encontradas são introduzidas com ocorrência igual a 1. Esta e a tabela **AIAD_WORDS** possuem os campos **code** (identificador da palavra), **word** (palavra) e **ocurrances** (nº de ocorrências da palavra).

2.4.2 Ficheiros das Frases

Simples ficheiros de texto que cada agente de pesquisa cria e escreve todas as frases encontradas na analise de uma página web. Após todos estes agentes terem terminado o agente principal procede ao agrupamento de todas as frases num ficheiro único que depois será utilizado para a aplicação do algoritmo de contextos. As palavras são guardadas em linhas diferentes por uma questão de eficiência e uma nova frase é detectada quando surge uma linha em branco.

2.4.3 Ficheiros dos Históricos dos Agentes

Tal como os ficheiros das frases também estes são simples ficheiros de texto que são criados por todos os agentes, quer os de pesquisa quer o principal. Cada linha de um ficheiro de histórico contém a hora a que a acção foi desenvolvida e uma descrição do que o agente executou.

Além das estruturas importa referir ainda sobre a implementação a comunicação existente entre os agentes que nossa aplicação se resume simplesmente a mensagens de terminação e erro por parte dos agentes de pesquisa para o agente principal. Através da mensagem o agente principal é capaz de terminar o agente de pesquisa se a mensagem recebida for de terminação ou de tentar resolver o problema se for uma mensagem de erro podendo terminar se não for possível a resolução.

2.5 Ambiente de Desenvolvimento

A aplicação foi desenvolvida maioritariamente em Java sendo que também foi utilizada a linguagem **mysql** para o acesso à base dados. Relativamente a plataformas começamos por utilizar o **netBeans** da **Sun** mas com o decorrer do tempo inúmeros problemas foram surgindo e optámos pelo simples editor **EditPlus** que nos permitiu ter um maior controlo no desenvolvimento do sistema. Os sistemas operativos utilizados foram o **Windows 2000** e o **Windows XP** instalados nos computadores da FEUP e em casa respectivamente. Em ambos os sistemas operativos recorremos à linha de comandos e à biblioteca Java instalada para compilação e execução do programa.

3. CONCLUSÃO

A implementação deste projecto permitiu-nos verificar este tema tem sido objecto de estudo por diversas pessoas e, como em quase tudo que diz respeito a inteligência artificial, não tem solução

ideal. Os maiores problemas prenderam-se com o facto de termos de implementar imensas coisas que nada têm a ver com a cadeira nem com o tema mas que eram essenciais para a realização do sistema de pesquisa tal como o acesso a base de dados **mysql** e a utilização da *API* do *Google*. Contudo podemos afirmar que o desenvolvimento do projecto foi feito com uma enorme motivação e que aquilo que não implementamos deveu-se bastante a falta de tempo e sobretudo a algum mau planeamento da nossa parte.

4. MELHORAMENTOS

A nível de melhoramentos ainda muito pode ser feita relativamente ao nosso projecto dada a sua complexidade e devido às diversas limitações existentes como a memória utilizada, a velocidade da rede e o tempo de processamento para a aplicação do algoritmo. No entanto, existem pequenos problemas que podem ser resolvidos como a detecção de frases numa página web, o algoritmo e a introdução de mais opções.

4.1.1 Detecção de Frases

Apesar de se tratar de uma parte bastante complexa da aplicação visto a dificuldade de encontrar frases numa página web ser imensa se tivermos em conta que uma página pode ser feita de diversas maneiras, por diversas pessoas e com diversos programas não seguindo portanto nenhum protocolo. Contudo o que nosso programa faz baseia-se na análise de caracteres de fim de frase (!?.;) não sendo portanto uma solução ideal já que elimina imensa informação. Seria necessário uma investigação pormenorizada para obter mais possibilidades de construção de frases e eliminar todas as ambiguidades presentes.

4.1.2 Algoritmo

Nas diversas aplicações que executamos sobre o nosso programa pudemos observar que na maior parte das vezes os contextos construídos não foram os mais correctos. Para melhorar este agrupamento a solução seria, em vez de apenas analisar as palavras que ainda não contém **cluster** em cada iteração, analisar todas e se encontrasse algum **cluster** onde o valor da fórmula fosse melhor a palavra seria removida do **cluster** antigo e inserido no novo. Seria também importante que o valor que limita o intervalo (**delta**) fosse controlado automaticamente pela aplicação já que este pode variar imenso com o número de palavras analisadas e com mais alguns factores.

4.1.3 Mais Opções

Apesar de constarem da interface da aplicação o sistema não é capaz de guardar uma pesquisa para posterior utilização. Esta componente é de fácil implementação e não foi realizada por falta de tempo da nossa parte.

5. REFERÊNCIAS

[1] API do Google

http://www.google.com/apis

[2] HTML Parser

http://htmlparser.sourceforge.net/

http://www.quiotix.com/downloads/html-parser/

[3] Word Clustering

http://pespmc1.vub.ac.be/CLUSTERW.html

http://www.di.ubi.pt/~pln/

http://tahoe.inesc-id.pt/

[4] Jade

http://sharon.cselt.it/projects/jade/

[5] Java

http://java.sun.com