4 Framework Proposto para Construção de Mediadores

Neste capitulo apresentamos um framework que implementa a estratégia para enriquecimento de dados a partir de informações da Deep Web, descrita no capítulo anterior.

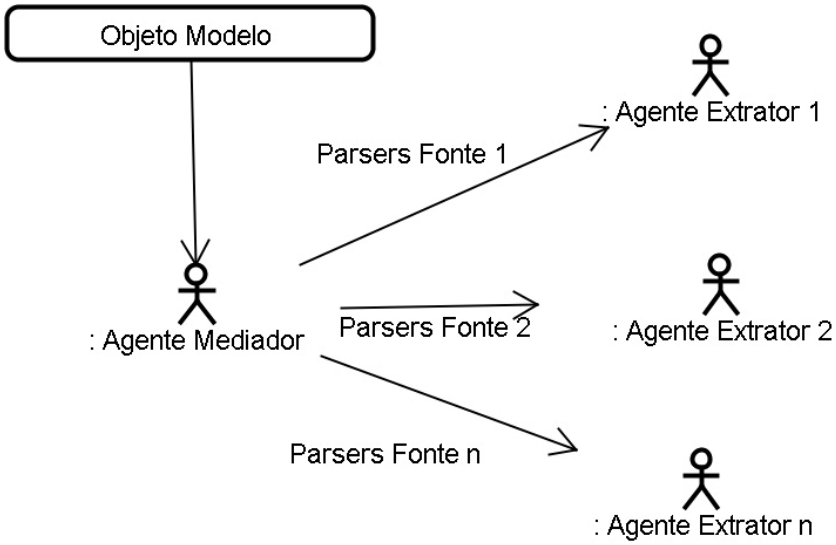
4.1 Visão Geral do Processo do Framework Proposto

O framework proposto implementa a estratégia para busca e incorporação de dados descrita no capítulo anterior. Seu funcionamento pode ser descrito por cinco passos fundamentais. Esses passos refinam os macro processos de busca (passos 1, 2 e 3) e incorporação (passos 4 e 5) da estratégia proposta.

Passo 1: Inicialização: Uma entidade modelo (tupla), parsers específicos de fontes de dados e parâmetros adicionais são carregados no mediador.
Nesta etapa o utilizador do framework já deve ter definido como diferenciar as os objetos do domínio do seu problema, deve ter definido também os valores de corte (cutoff) do classificador e limiar (threshold) das funções de similaridade escolhidas.

Passo 2: Distribuição: Para cada fonte de dados, um agente extrator é criado.
O utilizador deve ter codificado os parsers das páginas que são as fontes de dados. Cada agente extrator utilizará um conjunto de parsers relacionados (referentes a páginas do mesmo site). O utilizado deve ter implementado as possíveis conversões de dados que possam ser necessárias para integração futura dos dados no nível de esquema de dados assim como dos próprios dados.

Passo 3: Rastreamento de dados na Deep Web: Cada extrator realiza consultas sobre sua fonte diretamente em URL's ou formulários web. Uma vez com a página de resultados, ele navega pelos links, capturando informações sobre os candidatos a duplicata. O utilizador já deve ter implementado uma estratégia de montagem de consultas que atende aos padrões descobertos de URL's das fontes de dados.



(a)

(b)

Figura 13 - Passos 1, 2 (a) e 3 (b) do processo (estratégia de busca)

É necessário observar que neste processo o framework não deve obter recursos de uma mesma fonte em um intervalo pequeno de tempo, como 5 segundos, para evitar esgotar os recursos do site que hospeda as informações-alvo. Além disso, este não deve obter recursos de fontes que não autorizam a captura de dados automatizada.

Passo 4: Resolução de Entidades (Entity Matching): Cada agente extrator retorna seus candidatos a duplicata ao mediador. Após este passo o mediador treina o classificador com o modelo original e os dados das duplicatas. O classificador é treinado online de acordo com o algoritmo previamente apresentado, utilizando uma função de cálculo de similaridade que compara os candidatos com o modelo (tupla). Um candidato é considerado positivo se o valor retornado pela função for maior que um limiar (threshold) fixo, caso contrário é considerado negativo.

O classificador *default* escolhido para este framework é um classificador baseado no Vector Space Model descrito em [26]. Simples e eficiente, este classificador utiliza a as freqüências dos termos com pesos, em vez de probabilidades. Este tipo de classificador apresenta vantagens quando comparados aos classificadores Bayesianos. Como os pesos não são binários, é possível obter graus de similaridade de forma contínua, possibilitando ordenar os itens classificados. No entanto esta abordagem apresenta algumas limitações como em casos de comparação de cadeia de caracteres grandes e buscas exatas.

O classificador escolhido separa os itens em duas classes, aqueles com o valor maior ou igual ao valor de corte (*cutoff*) e os abaixo deste.

Ambos os parâmetros *cutoff e threshold* devem ser determinados previamente pelo utilizador do framework assim como a função de cálculo de similaridade entre cadeias de caracteres. Estas escolhas são baseadas no domínio dos dados.

Passo 5: Fusão dos dados: O agente mediador mescla as informações adicionais obtidas de acordo com a estratégia utilizada pelo utilizador do framework. Um exemplo de estratégia é simplesmente fundir informações que não estão nos registros originais do modelo, tais como preço e imagem, a partir das duplicatas descobertas.

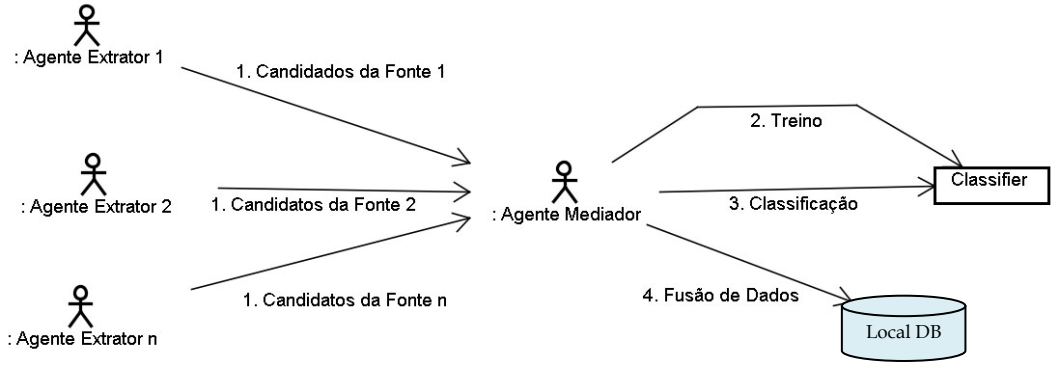


Figura 14 - Passos 4 e 5 do processo (incorporação)

4.2 Arquitetura Proposta

Dado que para cada domínio existem necessidades de personalização das etapas do processo foi desenvolvido um framework. O utilizador deve criar classes que estendem classes principais e sobrescrever alguns métodos. Além disso, deve realizar algumas configurações (via arquivo de configuração ou sobrescrevendo variáveis) antes da execução do processo.

O framework foi desenvolvido utilizando-se a linguagem Java dados os mecanismos do paradigma de orientação a objetos que facilitam a construção de frameworks e facilidade de obtenção de *frameworks* e API's *open-souce* que apóiam a construção deste sistema.

Toda documentação referente às classes do framework e suas instâncias está registrada no formato *Javadoc*, bem como estão presentes da pasta "docs/javadoc" do projeto e podem ser acessadas através do link http://www.winetag.com.br/scraper/.

As classes principais do framework estão no pacote br.com.scraper. As classes referentes a agentes estão no sub-pacote "agents". Até o fim desta versão não há instanciação *default* de nenhuma das classes abstratas. Na Figura 15, a seguir, apresentamos o diagrama de classes do framework proposto.

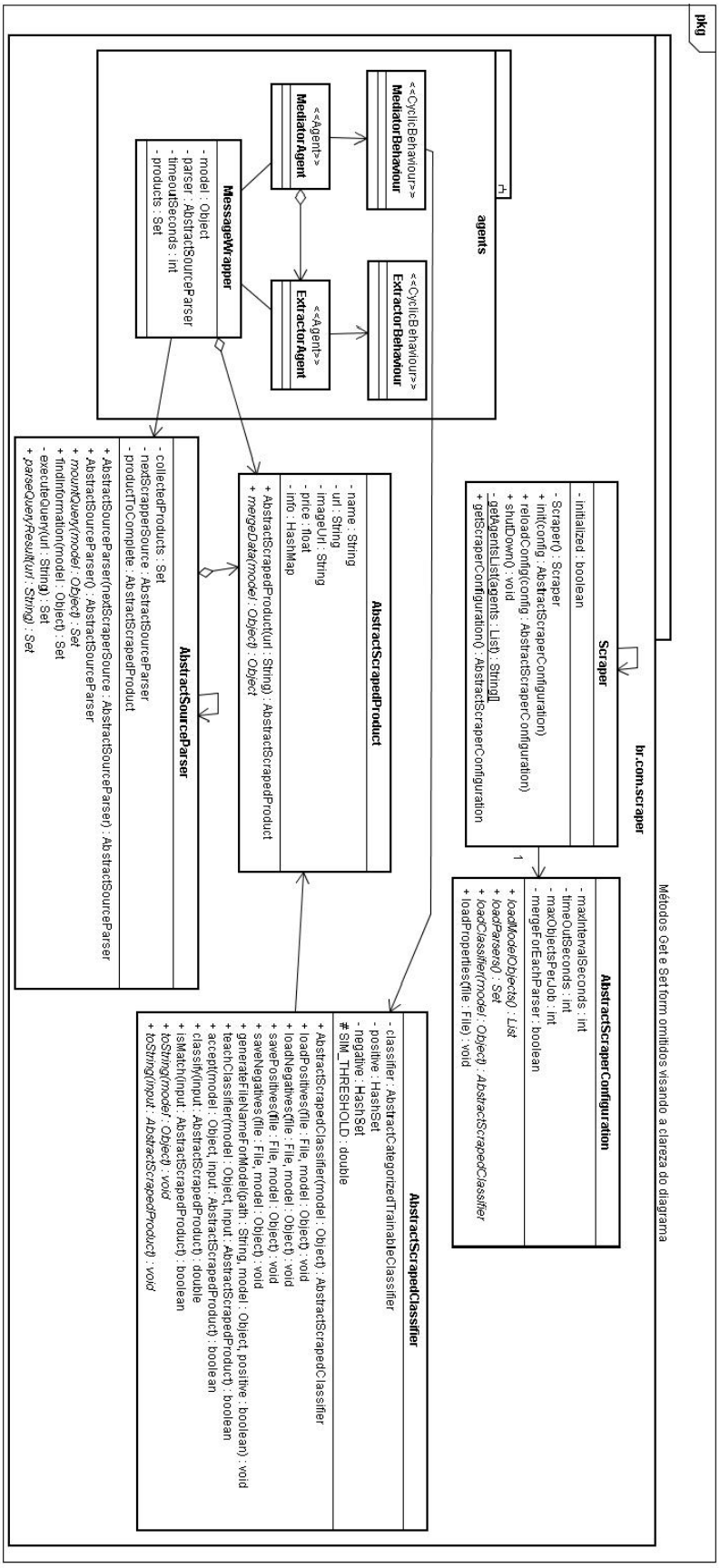


Figura 15 - Diagrama de Classes

A classe Scraper é a responsável pelo controle da execução de todo o processo. Ela configura, inicia e desliga os agentes que fazem a extração, transformação e carregamento dos dados. Scraper é uma classe que segue o padrão *Singleton* e deve ser inicializada com um objeto de configuração. É possível utilizar o framework sem esta classe, no entanto, sem o comportamento esperado.

A classe abstrata AbstractScraperConfiguration é responsável por armazenar as configurações do framework. Ela deve conter informações sobre o tempo máximo de intervalo entre cada extração (maxIntervalSeconds), o tempo máximo de espera pela resposta de um agente de extração (timeOutSeconds), a quantidade máxima de objetos a procurar por execução (maxObjectsPerJob) e se o framework deve realizar a fusão dos dados para cada *parser*, ou seja para cada fonte, ou para todas escolhendo o melhor candidato. As classes que estendem esta devem sobrescrever os métodos loadModelObjects, retornando os produtos que devem ser utilizados nas buscas, loadParsers, retornando os *parsers* responsáveis por interpretar as páginas web retornadas nas buscas e loadClassifier, retornando um classificador apropriado ao objeto que está sendo procurado.

A classe abstrata AbstractScrapedProduct representa os produtos encontrados nas buscas realizadas. Tem como atributos principais, comuns à maioria dos produtos disponíveis na internet, nome, URL da imagem, preço, informações. O atributo URL é o atributo principal e aquele que diferencia os produtos já que cada recurso da Web é acessado univocamente através de sua URL. As classes que estendem esta devem ser capazes de saber como realizar a operação de fusão, sobrescrevendo o método mergeData.

A classe abstrata AbstractSourceParser é responsável por gerar as URLs de busca e realizar o parsing dos resultados. É possível concatenar classes (*parsers*) desse tipo passando outras instâncias como parâmetro. A classe guarda os produtos coletados em collectedProducts, e a instância a ser chamada em caso de concatenação em nextScraperSource. É possível passar productToComplete para esta nova instância a fim de completar o produto já coletado com novas informações. As classes que estendem esta devem sobrescrever o método mountQuery retornando uma lista de URLs a serem exploradas e parseQueryResult, retornando os produtos coletados. A atividade de exploração dos links (*crawling*) é realizada pelo método executeQuery.

A classe abstrata AbstractScrapedClassifier é a interface para um classificador a ser utilizado na instância do framework. As classes que estendem esta devem sobrescrever os métodos toString para o modelo e o produto coletado gerando uma cadeia de caracteres que seja capaz de identificar cada um destes. É necessário também sobrescrever o método accept retornando se um modelo pode ser dito similar a um produto. O método generateFileNameForModel é utilizado para resolver o correto local de armazenamento dos arquivos temporários do classificador. Os métodos classify e isMatch retornam o grau de similaridade entre os produtos após utilizar o classificador treinado. O método teachClassifier ensina o classificador de acordo com a resposta dada pelo método accept, logo este é o método que deve ser sobrescrito com maior atenção. O método accept, que pode ser sobrescrito, está implementado como padrão para comparar as duas cadeia de caracteres geradas pelos métodos toString, utilizando como limiar o valor SIM_THRESHOLD de 90%.

Foi escolhido utilizar um classificador baseado no Vector Space Model, que é similar a um classificador Bayesiano como classificador *default*.

Os agentes contidos no framework manipulam as classes descritas de maneira correta, de acordo com as configurações dadas. O fato do sistema ser multiagentes diminui o tempo total de procura por um determinado produto já que é possível realizar buscas em diversos sites ao mesmo tempo.

MediatorAgent é o agente que centraliza as operações. Ele cria agentes do tipo ExtractorAgent para cada AbstractSourceParser disponível. Os agentes utilizam instâncias de MessageWrapper para facilitar a troca de objetos dentro das mensagens. MediatorAgent ainda treina e classifica os produtos encontrados pelos agentes extratores afim de encontrar os mais similares.

O Diagrama de componentes apresenta as dependências externas do framework. Estas bibliotecas devem ser incluídas nos projetos que o utilizarão. Pode-se ver que é utilizado o framework JADE para apoio a criação de agentes, a API Classifier4J para a criação de classificadores e as API's Simmetrics e SecondString para cálculo de similaridade entre cadeias de caracteres.

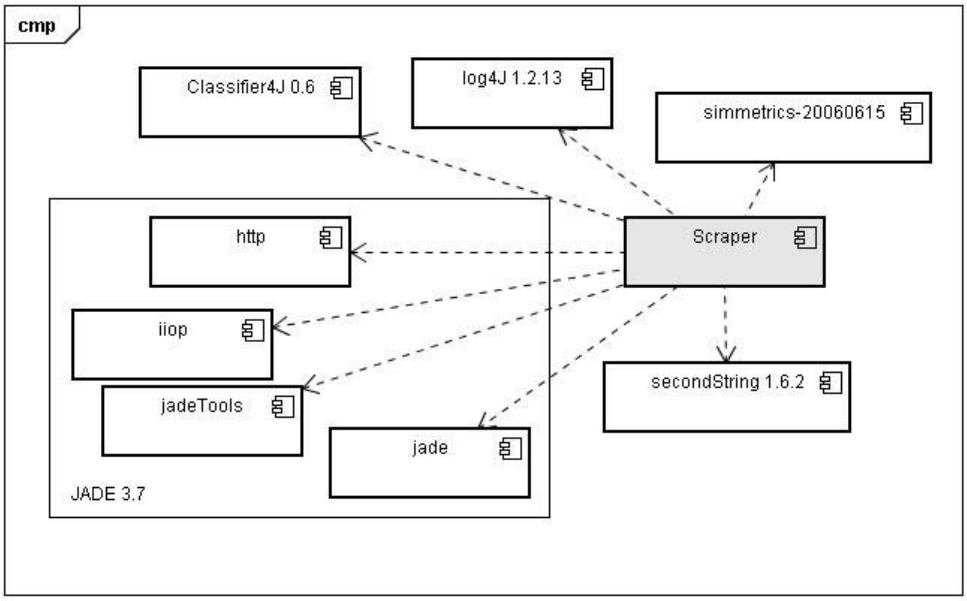


Figura 16 - Diagrama de Componentes do Framework

4.3 Resumo

Neste capítulo foi apresentada a solução codificada que implementa a estratégia de enriquecimento de dados proposta no capítulo anterior. São discutidos em detalhes aspectos relacionados ao código, e à instanciação do framework, para um domínio qualquer.

A partir da apresentação do processo de enriquecimento de dados, discutimos detalhes de implementação e funcionamento do framework computacional proposto, tais como as responsabilidades dadas a cada agente de software na realização de atividades ligadas à extração de informações e na mediação de dados. De forma a facilitar a implementação, decompomos a estratégia proposta no Capítulo 3 em cinco passos, a saber: (1) Inicialização, (2) Distribuição, (3) Rastreamento de Informações na Deep Web, (4) Resolução de Entidades e (5) Fusão de dados (Data Fusion). O macro processo de busca envolve os passos 1, 2 e 3, enquanto que o de incorporação envolve os processos 4 e 5.