Departamento de Engenharia Informática
Mestrado em Engenharia Informática – Computação Móvel

Redes Cognitivas

Ano letivo 2012/2013

Integração Semântica

Relatório

Documento elaborado por:
Cláudio Esperança – aluno n.º 2120917
Diogo Serra – aluno n.º 2120915

Docentes:
Vítor Basto Fernandes

Versão 1.0.27

Revisões

Versão	Autor(es)	Descrição das alterações
1.0	Cláudio Esperança Diogo Serra	• Versão inicial do documento

Índice de conteúdos

Índice de imagens

Lista de acrónimos

Para referência deixamos uma lista de acrónimos utilizados ao longo do documento:

Acrónimos	Descrição das alterações
ANTLR	*ANother Tool for Language Recognition*
API	*Application Programming Interface*
DEI	Departamento de Engenharia Informática
ESTG	Escola Superior de Tecnologia e Gestão de Leiria
GPL	*GNU General Public License*
HTML	*Hyper Text Markup Language*
IPL	Instituto Politécnico de Leiria
MEI	Mestrado em Engenharia Informática
MEI-CM	Mestrado em Engenharia Informática – Computação Móvel
UC	Unidade Curricular
UI	*User Interface*
XML	*eXtensible Markup Language*

1. Introdução

Este documento apresenta o trabalho desenvolvido no âmbito do projeto da unidade curricular de Redes Cognitivas do Mestrado em Engenharia Informática – Computação Móvel, da Escola Superior de Tecnologia e Gestão de Leiria no ano letivo de 2012/2013.

A singularidade e criatividade de cada individuo manifesta-se na forma como este produz o seu conhecimento e o partilha com os outros. Milhares de anos de convivência em sociedade fizeram com que o ser humano seja muito adaptável e tenha uma grande capacidade de interpretar, compreender e assimilar conceitos de outras pessoas. Infelizmente os sistemas de informação são (ainda) um pouco mais limitados e preferem estruturas formais mais normalizadas que permitam facilitar a interpretação e manipulação da informação.

Com o aparecimento e massificação da Internet foi criada a maior biblioteca de conhecimento do mundo. No entanto os sistemas de informação têm imensa dificuldade em interpretar esta informação pois esta não assume uma estrutura normalizada comum, que permita aplicar um algoritmo simples para estabelecer relações entre cada um dos conceitos. É neste contexto que surge o conceito de Web Semântica, como uma extensão da web atual que permitirá interligar significados de palavras, dando um sentido aos conteúdos, de modo a que estes sejam percetíveis e possam ser utilizados tanto por humanos como por computadores.

Este projeto enquadrou-se na área de estudo da Web Semântica, onde se pretendia a integração de bases de dados heterogéneos através da extração, processamento, análise e associação de caraterísticas obtidas a partir de descrições de recursos em linguagem natural. De um modo mais simples, pretendia-se o desenvolvimento de um mecanismo de identificação e associação automática (mapeamento) de

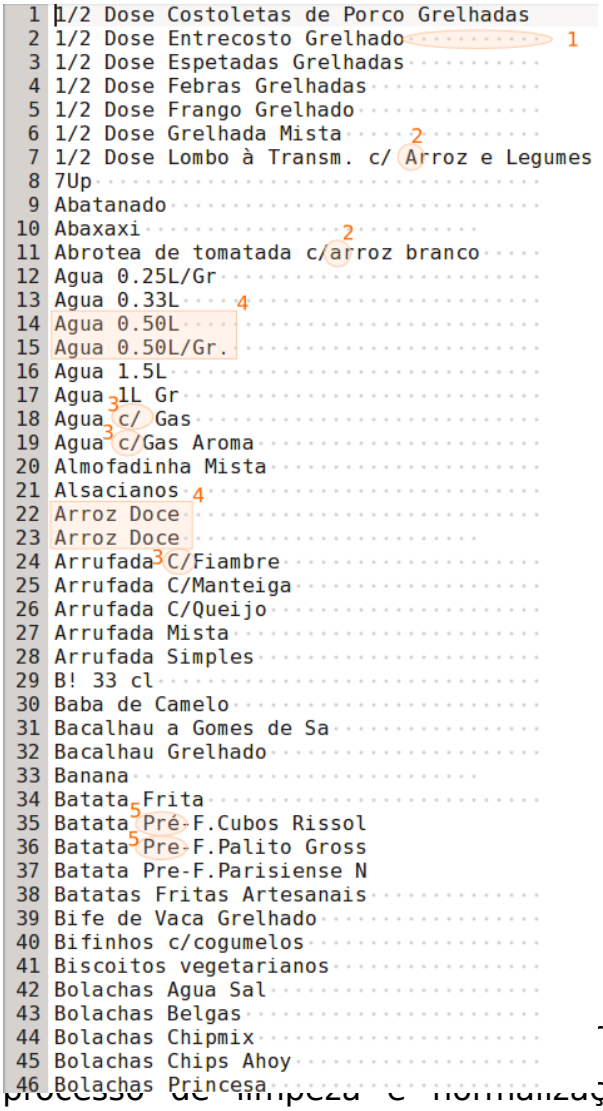
recursos (produtos de consumo alimentar) entre sistemas diferentes. Como as estruturas de dados que caraterizam cada um dos produtos em cada um dos sistemas são diferentes, pretendia-se que o mapeamento de produtos fosse feito com recurso à interpretação da descrição de cada um dos produtos (que poderia ser diferente nos dois sistemas para um mesmo produto).

Neste documento começaremos por apresentar, no capítulo 2, os resultados da análise e a solução proposta. Os detalhes da implementação serão apresentados no capítulo 3 e as habituais conclusões no capítulo 4.

2. Solução proposta

2.1. Análise da base de dados

O processo de análise começou com o estudo da base de dados para perceber o tipo e estruturas de dados que era necessário relacionar.



Verificou-se que cada linha do documento correspondia a um produto e onde eram mencionadas informações sobre quantidade, formato, ingredientes, etc. Constatou-se ainda a existência de alguns erros e inconsistências de dados, como por exemplo:

1. Existência de espaços
2. Inconsistência nas maiúsculas/minúsculas para o mesmo substantivo
3. Inconsistências no formato de algumas palavras
4. Existência de produtos duplicados
5. Inconsistências de ortografia

Verificou-se assim que antes de aplicar ansformação de dados, teria-se de aplicar um processo de limpeza e normalização dos dados para minimizar o efeito de

inconsistências.

2.2. Fases do processo

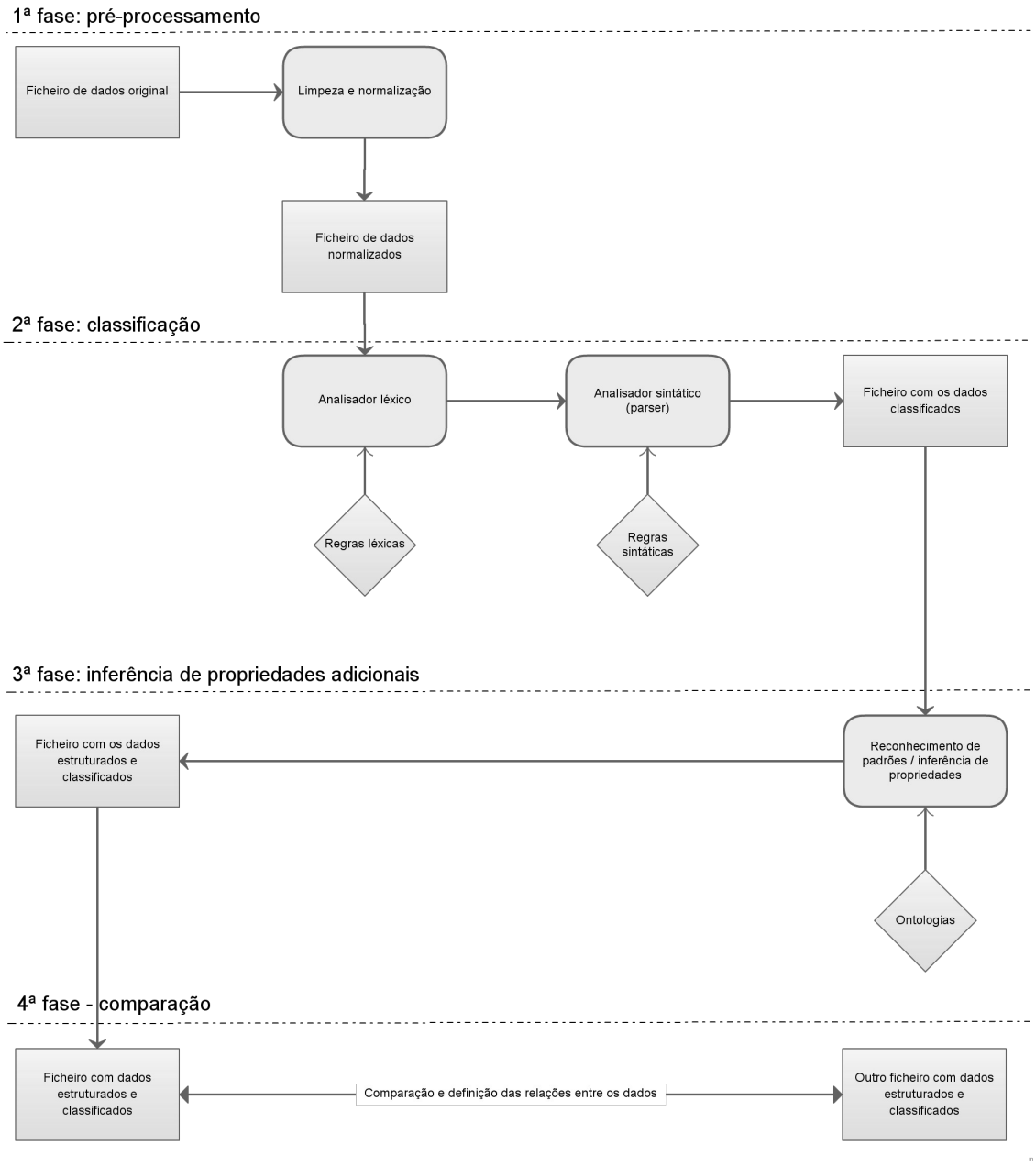


Imagem 2: Fases do processo de classificação e relacionamento de produtos

2.2.1. Limpeza e normalização

Após o carregamento do ficheiro de dados-fonte para memória, deve proceder-se a uma limpeza e normalização dos conteúdos do mesmo para minimizar os efeito de alguns dos problemas detetados na análise de dados, e melhorar a qualidade da informação.

Neste processo devem ser aplicadas as seguintes operações:

- Conversão de todos os carateres para minúsculas, minimizando assim inconsistências nos dados (por exemplo, “Arroz” e “arroz”, passam a ser o mesmo produto).
- Remoção de todos os diacríticos (com exceção do acento grave no “à”). Os acentos e a omissão de cedilhas são alguns dos erros mais comuns na digitação de texto no nosso idioma. A remoção destes símbolos gráficos permite minimizar o efeito deste erros ortográficos. Foi mantido o caracter “à” pois este indica um formato ou especificidade de um produto (por exemplo, “lombo à transmontana”).
- Adição do espaço após a abreviatura “c/”, para que esta abreviatura não seja associada à palavra seguinte caso não exista um espaço entre os dois elementos.
- Remoção de todos os espaços duplicados da frase, pois esta duplicação não adiciona qualquer valor semântico à frase e ao produto.
- Remoção de todas as linhas vazias.
- Remoção de espaços do inicio e final de descrições/frases.
- Remoção de todas as linhas duplicadas (as repetições não têm valor semântico)

2.2.2. Classificação

Nesta fase os dados deverão ser processados para que, com base num conjunto de regras sintáticas gramaticais da linguagem, sejam extraídas as propriedades relativas a cada um dos registos. Por exemplo, para um produto com a descrição "1/2 dose entrecosto grelhado" poderá ser obtido um resultado como o apresentado na imagem 3.

```
<DESCRICAO_COMPOSTA valor="1/2 dose entrecosto grelhado">
    <DESCRICAO_SIMPLES>
      <ALGEBRICO_FRACAO valor="1/2"/>
      <PROPRIEDADE_GRANDEZAUNITARIA valor="dose"/>
      <PROPRIEDADE_PARTEDEANIMAL valor="entrecosto"/>
      <PROPRIEDADE_CONFECAO valor="grelhado"/>
    </DESCRICAO_SIMPLES>
</DESCRICAO_COMPOSTA>
```

No exemplo da imagem anterior, um produto com a descrição mencionada foi identificado como sendo uma parte de um animal com o nome "entrecosto", cuja confeção é grelhada, servido em doses e cuja a quantidade é meia dose.

As regras sintáticas e gramaticais a definir dependerão sempre do domínio do conhecimento e da linguagem em que a informação está descrita. A sintaxe e gramática de um mesmo produto descrito em outro idioma poderá ser totalmente diferente, pelo que existe a necessidade de adaptar estas regras em função das necessidades.

Este processo de classificação, extração de propriedades e de transformação deve ser repetido para todos os produtos da base de dados para que os mesmos possam ser comparados com os de outra base de dados cujos produtos tenham sido submetidos ao mesmo processo.

2.2.3. Inferência de propriedades adicionais

Apesar de a associação de produtos ser já possível com as propriedades extraídas a

partir da análise sintática e gramatical da descrição dos produtos, é possível melhorar o grau de certeza dessa associação e os resultados obtidos através da inferência de propriedades adicionais sobre cada um dos produtos. Esta inferência é feita com recurso a ontologias que representam um conjunto de conceitos dentro de um domínio do conhecimento e os relacionamentos entre estes.

Com recurso a uma base de dados de conhecimento aplicada ao domínio do problema, e com base nas caraterísticas já conhecidas sobre um dado produto, será possível associar esse produto a classes e caraterísticas, permitindo utilizar as relações definidas nessa base de dados para inferir novas propriedades que não existiam inicialmente e, por isso, não foram obtidas diretamente a partir dos dados-fonte.

Como exemplo temos o caso do produto que sabemos que é um líquido (por exemplo, pela existência de uma unidade de grandeza líquida como é o caso de “cl” ou “l”), de cor “tinto”. Com base nestas caraterísticas, a base de conhecimento pode permitir inferir que este produto pertence à classe “vinho”, que por sua vez é uma “bebida alcoólica”.

As propriedades adicionais que são obtidas através deste processo permitem aumentar o número de propriedades comparáveis para a próxima fase, aproximando ou afastando dois produtos através da existência de mais ou de menos propriedades em comum.

2.2.4. Comparação

Nesta fase do processo, o produto selecionado deve ser comparado com os produtos da segunda base de dados no sentido de encontrar produtos cujas caraterísticas se aproximem do produto selecionado. Neste caso são analisadas cada uma das propriedades (obtidas direta ou indiretamente, respetivamente, a partir da análise sintático-gramatical ou através de inferência) e respetivos valores; quantas mais propriedades com valores iguais existirem em comum, mais próximos são os produtos, permitindo afirmar com maior grau de certeza que se trata do

mesmo produto.

Neste fase podem ser atribuídos pesos, dando mais relevância a uma propriedade relativamente a outra, no processo de comparação. Estes pesos devem ser ajustados de acordo com o domínio e especificidades do problema em estudo.

3. Implementação

3.1. Estudo de soluções de software

Dada a dimensão e especificidades das fases de classificação e inferência de propriedades adicionais, houve necessidade e encontrar soluções específicas, com vista à obtenção de um resposta eficaz para esses problemas. Assim, a classificação e extração de caraterísticas de produtos seria feita com ferramentas de reconhecimento de linguagens e as ontologias para a base de conhecimento editadas com uma ferramenta de edição de ontologias.

3.1.1. Ferramentas de reconhecimento de linguagens

Foram investigadas várias ferramentas, nomeadamente o *Lemon*[1], ANTLR[2], *Bison*[3], *Yacc/Lex*[4], *Ply*[5], *Jflex*[6] e *Quex*[7]. Após a análise de aspetos como funcionalidades, sintaxe dos ficheiros de definição das regras, utilitários, licença de uso, estado do projeto, linguagem base, complexidade, opiniões da comunidade e curva de aprendizagem, foram escolhidas duas ferramentas para uma análise mais aprofundada: ANTLR e *Yacc/Lex*.

O *Yacc* é geralmente utilizado em conjunto com o *Lex*, sendo que o último funciona como um pré-processador do primeiro, gerando os *tokens* (com base nas regras

1 http://hwaci.com/sw/lemon/
2 http://antlr.org
3 http://gnu.org/software/bison/
4 http://dinosaur.compilertools.net/yacc/
5 http://dabeaz.com/ply/
6 http://jflex.de/
7 http://quex.sourceforge.net/

léxicas) que serão utilizados pelo *Yacc* (em conjunto com as regras gramaticais fornecidas) para construir uma árvore sintática da estrutura que representa um dado registo. O *Yacc* tem sido substituído por outras soluções mais recentes como, por exemplo, o *Bison*. Após a análise desta solução, considerou-se a mesma complexa, com uma curva de aprendizagem elevada, e que obrigava a um forte conhecimento do funcionamento interno da aplicação para obter o tipo de dados pretendido.

O ANTLR apresentou-se como uma solução mais moderna, capaz de executar a análise sintática e gramatical de forma integrada e não dependo de outras soluções. Permite a existência de uma notação única e consistente, tanto para análise léxica, como a sintática, facilitando o seu uso. Fornece ferramentas gráficas para a edição e validação das regras, integradas no seu próprio ambiente de desenvolvimento ou em outros ambientes de desenvolvimento mais populares através de extensões. É uma solução com uma comunidade de desenvolvimento ativa com lançamentos regulares, como o que aconteceu no início deste ano. A sua utilização é gratuita e a solução é disponibilizada sobre uma licença BSD. Existe alguma documentação sobre a solução, incluindo alguns livros.

O ANTLR foi a solução escolhida como ferramenta de conhecimento de linguagem.

3.1.2. Ferramentas de edição de ontologias

Foram investigadas várias ferramentas para a edição de ontologias[8] (tais como o *Altova SemanticWorks*, *Amine*, *Apelon* DTS), mas a escolha recaiu sobre o *Protégé*. Esta ferramenta tem uma enorme comunidade de utilizadores registados (quase 200000[9]) e é disponibilizada sob uma licença de código aberto.

8 http://techwiki.openstructs.org/index.php/Ontology_Tools

9 http://protege.stanford.edu/reg-data/registration-data.html

3.2. Construção das regras sintáticas e gramaticais

A definição das regras sintático-gramaticais demonstrou ser um desafio complexo que implicaria uma análise demorada de cada registo da base de dados fornecida para a definição da sintaxe e gramática que permitisse a extração e transformação dos dados. Felizmente foram fornecidos pelo docente da unidade curricular alguns exemplos de gramáticas e sintaxes utilizadas no *Lex/Yacc* que foram convertidas, adaptadas e ajustadas para o problema em estudo e para um formato compatível com o ANTLR. A transformação das regras fornecidas em *Lex/Yacc* foi efetuada sem problemas de maior. No entanto foi necessário investigar a melhor forma de transformar algumas das expressões regulares utilizadas no formato fornecido para as expressões regulares que o ANTLR utiliza.

A ferramenta *Syntax Diagram* do ANTLR desempenhou um papel fundamental na adaptação e construção das regras, permitindo uma rápida perceção de como cada uma das regras vai evoluindo, facilitando o processo de definição da mesma. Isto permitiu acelerar consideravelmente este processo, quando comparado ao método anterior onde utilizámos um sistema de desenvolvimento por tentativa e erro (o que fez perder algum tempo precioso nesta primeira fase do processo).

Após a definição das regras, estas foram testadas com o *TestRig* do ANTLR, solução que permite a introdução de texto, ao qual são aplicadas as regras definidas; o resultado da aplicação dessas regras no texto fornecido é depois mostrado, o que permite validar os resultados obtidos e proceder a correções, caso exista essa necessidade.

Depois das regras terem sido definidas e validades, foi necessário definir as ações que deveriam ser executadas pela biblioteca do ANTLR para transformar os resultados no formato pretendido. O objetivo final era o de gerar uma *string* com uma estrutura XML com cada um dos produtos e respetivas caraterísticas obtidas através da análise sintático-gramatical. Esta tarefa que parecia, à primeira vista,

relativamente simples revelou-se bastante complexa. A documentação sobre as *actions* para a versão 4 (última à data atual) do ANTLR era muito limitada, com exemplos escassos e pouco abrangentes. A gramática que foi desenhada no âmbito deste projeto continha regras com múltiplos *tokens* e, inclusivamente, recursividade entre regras. Esta recursividade levantou alguns problemas, dado que, na criação do XML, nas *actions* os atributos devolviam `null`, interrompendo a recursividade e fazendo com que a análise não passasse por todos os *tokens* previstos (contrariamente ao que acontecia anteriormente com o *TestRig*). Só depois de muita investigação foi possível, e através de um exemplo de uma gramática para a versão 2 do ANTLR, encontrar uma solução que permitia atribuir nomes diferentes às regras, de modo a utilizar esses mesmos nomes nas *actions*, o que permitiria que as regras continuassem e assim não quebrar a recursividade.

Por exemplo a regra,

```
a:a b {a.value }
     | b c {b.value}
;
```

Imagem 4: Exemplo de uma regra com problemas

não iria efetuar a recursividade, uma vez que a *action*, ao ter o `a.value` iria devolver `null`; para evitar esta situação teríamos de dar um nome ao `a`, por exemplo:

```
a : aLetter = a bLetter = b {aLetter.value}
     | bLetter = b cLetter = c {bLetter.value}
;
```

Imagem 5: Exemplo de uma solução ao problema da regra anterior

Assim a recursividade não é afetada e, ao ser invocada a propriedade `value` do atributo, este não devolve `null` e continua para a próxima regra. Após esta descoberta foi possível gerar um XML mais próximo do que era pretendido, com todas as palavras analisadas e classificadas.

Em suma, muito do tempo alocado para o projeto foi utilizado nestes ajustes da gramática quando, em teoria, seria um processo relativamente simples.

3.3. Construção da ontologia

Para a modelação da ontologia recorremos ao *protégé* para reprodução das classes e propriedades identificadas na fase anterior.

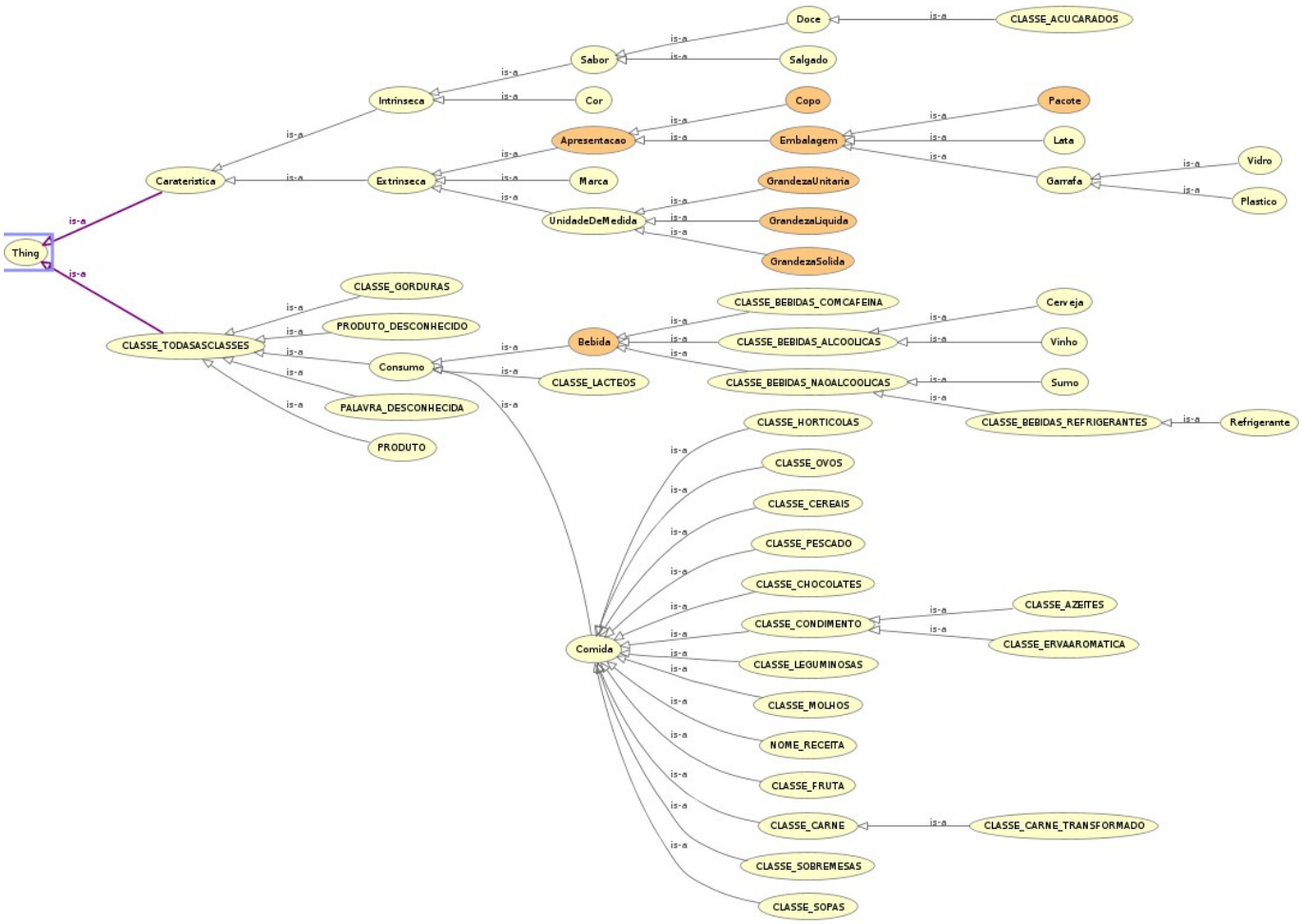


Imagem 6: Classes e respetivas relações na ontologia

As classes foram hierarquizadas e as propriedades foram relacionadas com as classes de modo a que, ao associar um produto a determinadas propriedades, este

possa ser classificado de acordo com o conhecimento existente na ontologia. Este conhecimento associa as classes às propriedades conhecidas do objeto e infere novas propriedades que podem ser associadas a esse objeto (por fazerem parte de classes aos quais o objeto foi relacionado).

3.4. Implementação do protótipo

O protótipo da solução foi implementado com recurso à linguagem de programação Java. A estrutura de ficheiros do protótipo encontra-se representada na imagem 1.

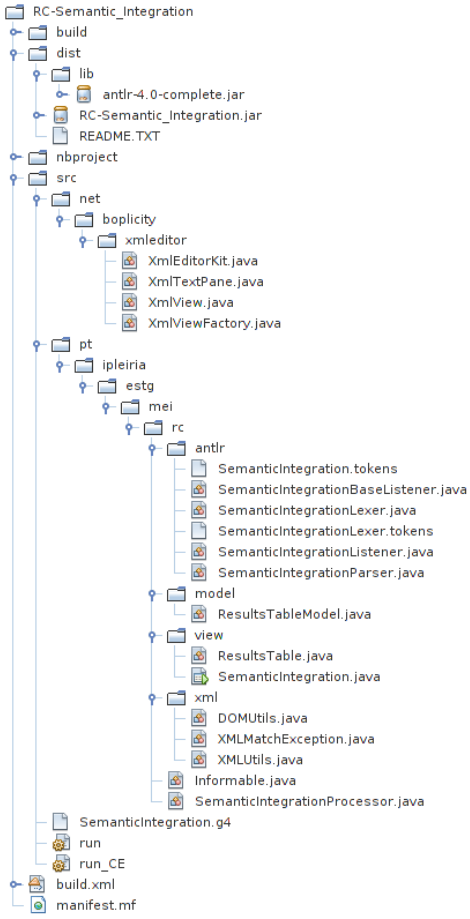


Imagem 7: Estrutura de ficheiros do protótipo

Na pasta `dist/` estão disponíveis os ficheiros de distribuição, incluindo o ficheiro `RC-Semantic_Integration.jar` que pode ser utilizado para executar o protótipo através do comando:

```
java -jar ./dist/RC-Semantic_Integration.jar
```

Em `src/net/boplicity/xmleditor`, estão disponíveis algumas classes utilizadas no GUI para a apresentação do XML num formato visualmente mais legível (com destaque de sintaxe).

O ficheiro `src/SemanticIntegration.g4` contem as regras sintático-gramaticais para geração das classes pelo ANTLR utilizadas na extração de caraterísticas a partir dos dados fornecidos. Os ficheiros `src/run` e `src/run_CE` são *scripts bash* com os comandos utilizados para gerar as classes mencionadas anteriormente e que ficarão disponíveis para utilização na aplicação

na pasta `src/pt/ipleiria/estg/mei/rc/antlr/`.

No ficheiro `src/pt/ipleiria/estg/mei/rc/view/SemanticIntegration.java` está definida a classe e os respetivos componentes que compõem o GUI da aplicação, bem como a função de inicialização da aplicação. O ficheiro `src/pt/ipleiria/estg/mei/rc/view/ResultsTable.java` contem a classe responsável pela vista da tabela de resultados e que utiliza o modelo definido na classe do ficheiro `src/pt/ipleiria/estg/mei/rc/model/ResultsTableModel.java`.

Na pasta `src/pt/ipleiria/estg/mei/rc/xml/` estão disponíveis alguns ficheiros com classes e métodos auxiliares para manipulação de conteúdos XML.

No ficheiro `src/pt/ipleiria/estg/mei/rc/SemanticIntegrationProcessor.java` está definida a classe que contem o algoritmo principal da aplicação responsável por, numa *thread* auxiliar carregar um ficheiro XML, invocar os métodos de limpeza e de normalização mencionados no capítulo 2.2.1, invocar os métodos auxiliares para processamento e transformação de dados para o formato XML final (com recurso às classes geradas pelo ANTLR), reformatação do XML gerado, avaliação da semelhança entre dois nós XML e notificação do interface gráfico das alterações de estado e do progresso das tarefas.

A utilização de uma *thread* independente para carregamento dos dados permite que possam ser carregados e processadas as duas base de dados em simultâneo, mantendo a fluidez e independência do GUI.

Por limitações de calendário, neste protótipo não foi implementada a API que faria uso do ficheiro OWL gerado na construção da ontologia e que poderia permitir inferir propriedades adicionais sobre cada um dos registos. Ainda assim, apenas na comparação direta entre propriedades obtidas diretamente a partir das bases de dados fornecidas, foi possível verificar que é possível encontrar detetar relações entre produtos de bases de dados distintas. Esta comparação e cálculo de semelhança é feita com base na igualdade dos atributos na raiz do nó (20%) e percentagem de semelhança dos nós filho (80%).

Na imagem 8 é apresentado o protótipo final com os resultados obtidos numa

pesquisa por um dado produto numa segunda base de dados.

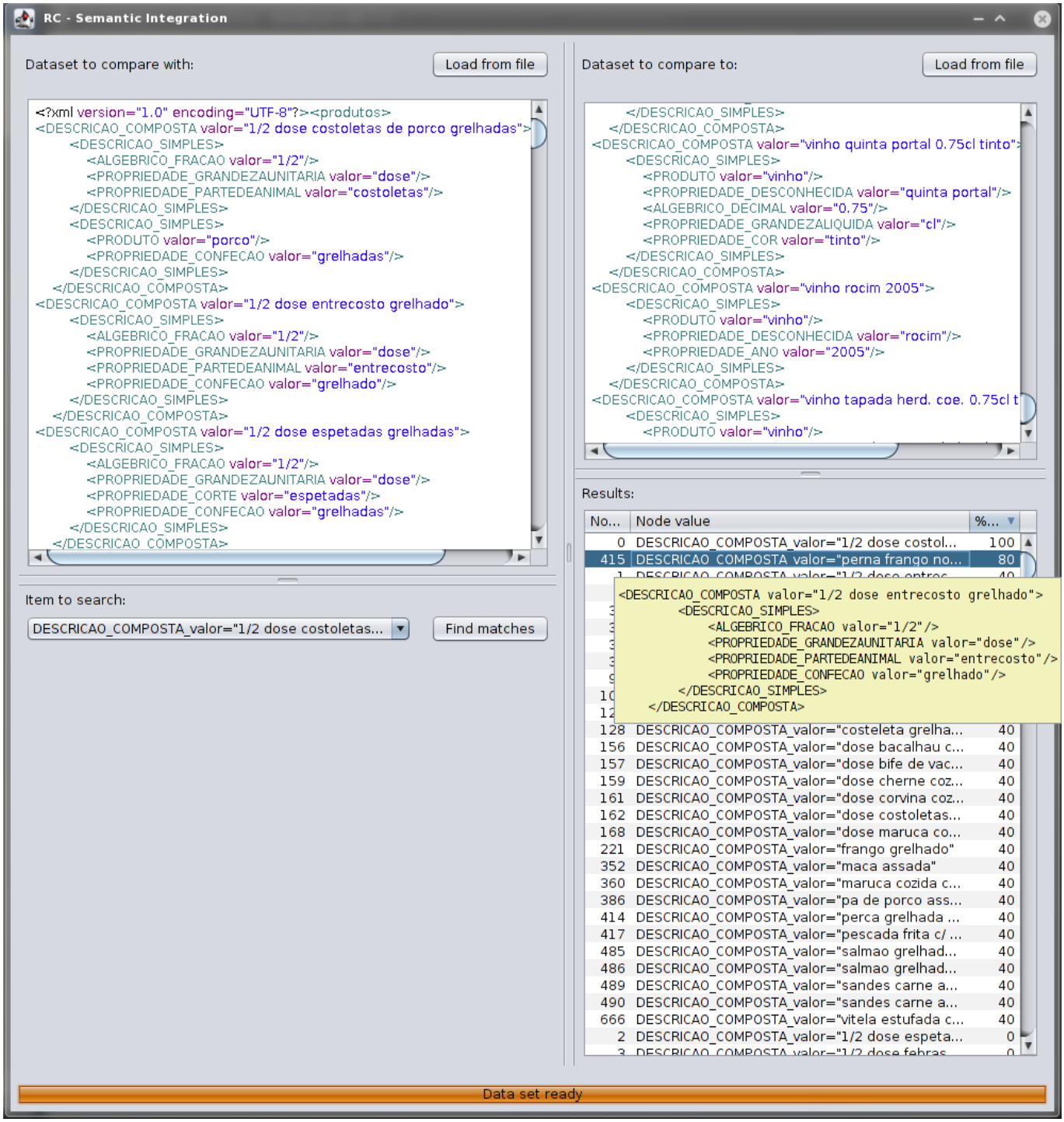


Imagem 8: Protótipo da aplicação

O processo de execução de uma pesquisa é relativamente simples:

1. Carregar o ficheiro com a base de dados de itens a pesquisar utilizando o botão “*Load from file*” no painel à esquerda
2. Carregar o ficheiro com a base de dados de itens a relacionar com o item a pesquisar utilizando o botão “*Load from file*” no painel à direita
3. Selecionar o registo a pesquisar na lista pendente no painel à esquerda
4. Utilizar o botão “*Find matches*” para pesquisar pelo itens semelhantes na base de dados à direita ao item selecionado
5. Ordenar os resultados na tabelas de resultados.

Ao deslocar o rato sob a coluna “*Node value*” no registo pretendido, deverá ser apresentado o XML do respetivo nó numa etiqueta.

As caixas onde o XML é apresentado são editáveis, sendo possível adicionar propriedades adicionais a um determinado elemento.

4. Conclusões e trabalhos futuros

A oportunidade de investigar tecnologias e problemas muito atuais tornaram este projeto num trabalho muito interessante. No entanto, a complexidade das temáticas envolvidas acabaram por comprometer um pouco a execução das 4 fases do projeto. De facto, se estudadas de modo mais aprofundado, qualquer uma destas fases daria um tema para um trabalho próprio.

De qualquer modo, pensamos que os objetivos deste projeto eram apenas os de um contacto introdutório a cada uma dos temas e não um estudo exaustivo sobre os mesmos. Acreditamos que estes objetivos foram atingidos, mesmo não se tendo conseguido implementar no protótipo final, as ontologias que foram definidas.

A escolha do ANTLR, que parecia ser a escolha mais acertada ao inicio (não só pelo seu rápido crescimento mas também pela muita documentação que parecia existir), revelou-se algo dececionante. Apesar de ser uma escolha válida e capaz, a curva de aprendizagem é bastante elevada e os exemplos existentes não focam alguns pontos essenciais (como as transformações com recurso às ações). O tempo que se dedicou à resolução dos problemas detetados acabou por comprometer as outras fases do projeto, em especial a integração das ontologias no protótipo final. Dado que a construção destas ontologias só poderia avançar depois da estrutura de dados resultante da transformação estar concluída, quando finalmente as ontologias estavam construídas, não existia tempo suficiente para a sua integração no protótipo.

No entanto, mesmo após estes problemas, acreditamos que deixamos uma base de trabalho sólida, que pode e deve ser continuada no futuro. A integração de uma API que permita utilizar ontologias para inferir novas propriedades é perfeitamente possível neste sistema que foi desenvolvido, bastando adicionar essas propriedades ao XML dos dados gerados pela aplicação para que as mesmas sejam

automaticamente utilizadas no processo de análise e de comparação.

O sistema de classificação também poderá ser modificado, permitindo ajustar os pesos de cada uma das propriedade no algoritmo para a obtenção de outro tipo de resultados.