Mestrado Engenharia Informática e Computação Móvel **2011-12** **1º Ano** **2º Semestre**

IPL escola superior de tecnologia e gestão instituto politécnico de leiria

Departamento Engenharia Informática http://www.dei.estg.ipleiria.pt/

Integração Semântica

Produtos Alimentares

Desenvolvido por:

César Ferreira	2111413@my.ipleiria.pt
Jóni Batista	2111416@my.ipleiria.pt

ÍNDICE

1 Introdução 4
2 Problema 5
3 Tecnologias e Ferramentas 6
3.1 Flex 6
3.2 Bison 6
3.3 Protégé 7
3.4 Java 7
3.5 APIS DE OWL 8
3.5.1 APIS DE OWL 8
3.5.2 HermiT 8
3.6 Linux/Bash 8
4 Desenvolvimento 9
4.1 Normalização dos Dados 9
4.2 Tratamento Léxico e Gramatical 10
4.3 Ontologia 12
4.4 Pesos 13
4.5 Matching dos Produtos 13
5 Funcionamento da Aplicação 16
6 Conclusões e Trabalho Futuros 17
7 Referências 18

1 INTRODUÇÃO

A necessidade em relacionar informação que se encontra dispersa por sistemas diferentes, constitui um problema recorrente da engenharia informática. Embora exista em grande quantidade informação disponível em suporte digital que seria útil e que teria muito valor se fosse relacionada/combinada, essa informação encontra-se tipicamente distribuída por diversos sistemas computacionais, com diferentes sistemas operativos, diferentes sistemas de base de dados, diferentes modelos de dados, diferentes formatos/representações/tipos de dados, diferentes formas de identificação dos recursos, dificultando muito a sua combinação e impedindo que se retire valor da sua existência em suporte digital.

Para que esta informação possa ser usada de forma combinada, é necessário criar modelos de conceitos e de relações ao nível do significado, por forma a construir processo de mapeamento automático entre a informação disponível em sistemas diferentes/heterogéneos.

A maioria das normas aplicadas para a interoperabilidade de sistemas especificam apenas o nível sintático da interoperação. Como os sistemas que partilham formas de identificação e sintaxes comuns são relativamente poucos (quando olhamos para o conjunto de serviços e aplicações presentes na Internet), a interoperabilidade entre sistemas heterogéneos poderá ser melhor conseguida à custa de uma camada protocolar semântica, suportada pelas normas propostas pela Web Semântica.

2 PROBLEMA

Pretende-se desenvolver um mecanismo de identificação automática de recursos (produtos) entre sistemas heterogéneos que usam formas de identificação (identificadores) diferentes para os seus recursos (produtos). O mecanismo a implementar deve permitir associar aos produtos existentes numa base de dados (que vamos designar de base de dados BD-A), as características adicionais presentes noutro sistema/noutra base de dados (que vamos designar de base de dados BD-B). Note-se que a forma como os produtos são identificados na BD-A é diferente da forma como os produtos são identificados na BD-B. A identificação de um produto da BD-a na BD-B, ou seja o processo de mapeamento, deve ser feito automaticamente pelo mecanismo a desenvolver, com base nas descrições textuais em português do produto na BD-A e na BD-B. Não existe garantia que a descrição textual de um mesmo produto na BD-A e na BD-B sejam exatamente iguais, pelo que o mapeamento deverá ser feito com base na semelhança de significado das descrições. Normas da Web Semântica deverão ser usadas para este efeito (OWL).

3 TECNOLOGIAS E FERRAMENTAS

Para implementar a nossa solução do problema apresentado foram utilizadas várias tecnologias e ferramentas, todas elas *open source*, descritas de seguida.

3.1 Flex

Flex é uma ferramenta de análise lexical que permite processar sequências de caracteres e converte-los em sequências de *tokens,* sendo uma alternativa ao Lex com distribuição gratuita. O Flex surgiu por volta de 1987, pela mão de Vern Paxson, e utiliza a linguagem C, sendo a última versão de 2008 [1].

O funcionamento dos programas escritos em flex é bastante simples. Os programas em Flex estão preparados para ler ficheiros passados por parâmetro, ou sequências de caracteres caso não correspondam ao nome de nenhum ficheiro. Na prática o programa irá fazer uma parse aos *inputs* através das expressões regulares e código C criado pelo programador, designadas de regras. Para isso é necessário compilar o código, gerando um ficheiro "lex.yy.c" e por sua vez o executável. Quando o programa é executado, este analisa as ocorrências de texto que correspondem às expressões regulares para cada regra. Sempre que encontra uma correspondência, ele executa o código C correspondente.

3.2 Bison

Bison é um *parser* para fins gerais que converte uma descrição gramatical de uma gramática livre de contexto LALR num programa C que analisa essa gramática. Assim que se esteja confortável com Bison, pode-se utiliza-lo para desenvolver uma vasta gama de *parsers* de linguagens, desde simples calculadoras até linguagens de programação complexas [2].

O Bison tem compatibilidade ascendente com Yacc: todas as gramáticas Yacc devidamente escritas devem funcionar com Bison sem alterações. Qualquer pessoa familiarizada com Yacc deve ser capaz de usar Bison com pouca dificuldade [3].

3.3 Protégé

O editor Protegé, originalmente desenvolvido pelo departamento de informática médica da universidade de Stanford e atualmente a ser mantido pela Universidade de Stanford em colaboração com a Universidade de Manchester, é considerado uma das melhores ferramentas de construção de ontologias, possibilitando a interação entre outros sistemas que utilizam base de conhecimento para resolver problemas de um determinado domínio [4].

Na sua infância, o Protegé era uma ferramenta de aquisição de conhecimento limitada a um Sistema Especialista para oncologia. Foi modernizado gradualmente para acompanhar a evolução da tecnologia de KBS (Knowledge-based systems). Permite criar formulários para a aquisição de conhecimento baseado em ontologias, edição de ontologias e knowledge-bases. Tem independência de um algoritmo específico de inferência, e fornece uma API de representação de conhecimento que permite a extensão do programa para necessidades específicas.

3.4 Java

Java é uma linguagem de programação orientada a objectos desenvolvida na década de 90 por uma equipa de programadores da Sun Microsystems, sendo adquirida pela Oracle anos mais tarde a quando compra da Sun pela Oracle.

Java foi utilizado como um complemento às ferramentas de parse (Bison e Flex), onde é efectuado uma pequena limpeza aos dados para remover acentos e caracteres especiais irrelevantes, antes de estes serem processados pelo Bison e Flex. Posteriormente é também utilizado para trabalhar com as APIs de ontologia escolhidas, sendo este último projecto o responsável por fazer o *match* dos produtos.

3.5 APIS DE OWL

Para a manipulação da ontologia de forma a inferir características foram usadas duas APIS em Java.

3.5.1 APIS DE OWL

A OWL API é uma API Java e implementação de referência para a criação, manipulação e serialização de ontologias OWL [5].

3.5.2 HermiT

É um *reasoner* para ontologias escrito usando a *Web Ontology Language*. Dado um ficheiro OWL, o HermiT consege determinar se uma ontologia é ou não consistente, assim como fazer a subsunção de relações entre as diversas classes, entre outras coisas [6].

3.6 Linux/Bash

Embora algumas das ferramentas por nós desenvolvidas durante a criação do nosso projeto sejam multi plataforma, as restantes tiveram que assentar sobre sistemas linux pois este é um ambiente muito mais *script-friendly* e dado que a ferramenta Bison é o gerador de *parsers* por omissão na grande generalidade dos sistemas unix utilizado em parseria com o Flex , temos assim garantia que tem a robustez necessária para desenvolver um projeto como o nosso.

4 DESENVOLVIMENTO

De forma a seleccionar o problema proposto desenhou-se um *workflow* de processos necessários para modelar o conjunto de tarefas a realizar e os diferentes intervenientes em cada processo. A figura 1 é a representação esquemática do *workflow*.

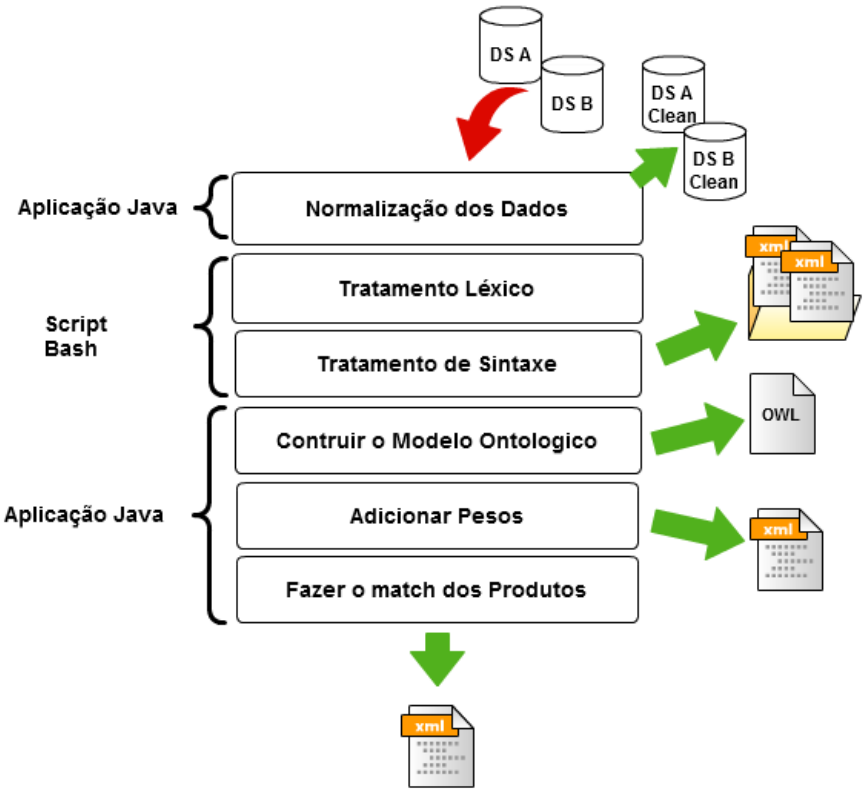

Figure 1 – Workflow do sistema

O esquema apresenta as várias etapas do projecto e o resultado esperado em cada uma delas. Os tópicos seguintes descrevem o que cada uma das etapas.

4.1 Normalização dos Dados

Um dos primeiros problemas com que nos deparamos ao tentar recolher os dados de diferentes sistemas heterogéneos é a disparidade de formas com que um produto é descrito, desde logo no que toca aos caracteres especiais, maiúsculas e minúsculas, etc.

Assim sendo, optou-se por criar uma aplicação em Java que processasse os dados de forma a fazer as seguintes modificações:

- Remover caracteres acentuados;
- Transformar caracteres maiúsculos para minúsculos;
- Remover caracteres especiais irrelevantes (só são válidas letras de “a” a “z”, “.”, “%” e “/”).

Este aplicação processa os *data sources* da BD-A e BD-B, executando as alterações descritas anteriormente através de expressões regulares. Por fim os dados normalizados, limpos, são guardados em novos ficheiros, prontos a serem consumidos pelo Bison/Flex.

Estas alterações poderiam ter sido realizadas no próprio *parser* lexical (Flex) utilizando o também expressões regulares para ignorar os caracteres referidos em cima, evitando assim criar um projeto com o único intuito de normalizar dados. O motivo pelo qual se optou por este caminho foi simplesmente para diminuir a complexidade das expressões regulares no Flex e evitar a repetição das mesmas ao longo dos diferentes *tokens*.

4.2 Tratamento Léxico e Gramatical

Após os *data sources* estarem normalizados é chegado o momento de os processar. O objectivo desta fase é identificar o domínio de cada vocábulo, *token,* e especificar a gramática/sintaxe que define um produto através de um conjunto de vocábulos. Ou seja, um produto é constituído por vários vocábulos relacionados gramaticalmente, onde cada vocábulo tem uma característica.

Sendo assim, era essencial interligar uma ferramenta lexical e gramatical, daí a escolha do Flex e Bison pois, estes dois interagem na perfeição. O funcionamento é bastante simples, o Bison irá enviar vocábulo a vocábulo para o Flex analisar e classificar, isto para todos os produtos. No nosso caso cada linha do ficheiro é um produto. O Flex classifica o *token* de acordo com as regras e devolver a sua caracterização para que o Bison transforme essas caracterizações numa descrição consoante a gramática e sintaxe definida através do conjunto de regras individuais usadas numa língua (cada língua têm

a sua própria gramática). As regras regem a construção frásica. A figura 2 é a representação esquemática do tratamento léxico e gramatical para um dado produto.

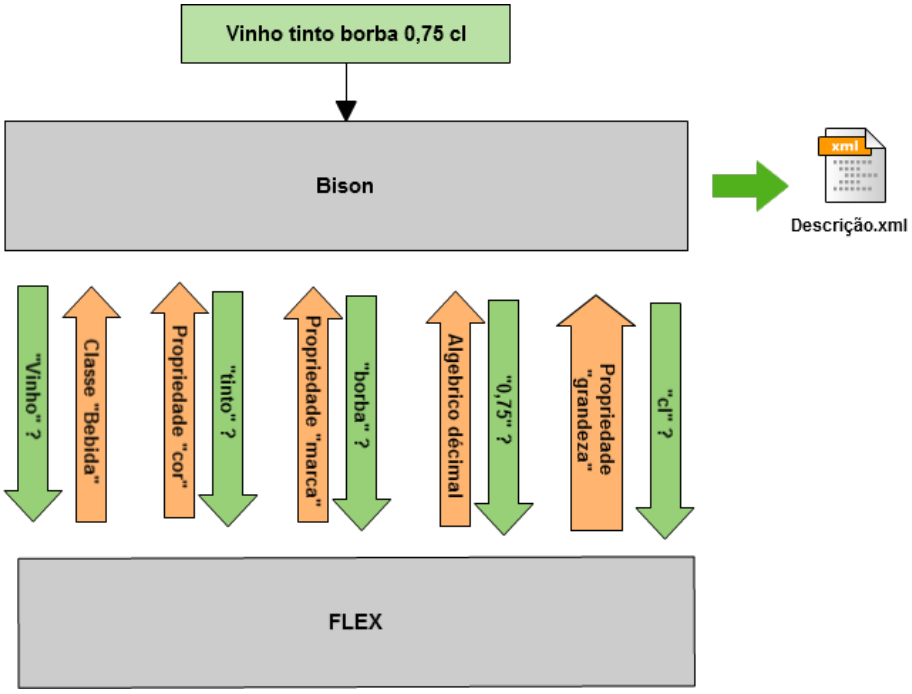

Figure 2 – Exemplo da interoperabilidade entre o Flex e o Bison

Como se pode observar o Flex identifica cada *token* isoladamente sendo depois da responsabilidade do Bison descrever o produto no seu conjunto. Gramaticalmente definiu-se que a descrição do produto é composta por classes, propriedades, quantidades, etc. As quantidades são compostas por uma grandeza e por um número. Por sua vez, um número pode ser decimal ou inteiro. Esta definição gramatical deveria ser feita para todo o domínio e forma a poder exprimir o máximo possível da linguagem. A figura 3 seria o output da descrição do produto exemplificado.

```
- <xml version="1.0" encoding="utf-8">
  – <linha>
      <produto familia="bebida">vinho</produto>
      <propriedade atributo="marca">borba</propriedade>
      <propriedade atributo="quantidade">0.75</propriedade>
      <propriedade atributo="grandeza">cl</propriedade>
      <propriedade atributo="cor">tinto</propriedade>
    </linha>
    ..
</xml>
```

Figure 3 - Resultado do processamento lexical e gramatical de um produto

O resultado final deste processo seria um ficheiro por *data source* em xml com a descrição de cada produto.

4.3 Ontologia

A Ontologia é uma ciência que permite a extração de informação. O objectivo da aprendizagem ontológica é extrair conceitos relevantes de forma semiautomática. Por outras palavras, a utilização de modelos ontológicos permito-nos acrescentar informação por inferência. Este processo é extremamente importante para colmatar os casos em que a informação oferecida pelos sistemas heterógenos é insuficiente para fazer o *match* dos produtos de forma acertada e precisa apenas com base na análise lexical e gramatical. Por exemplo, se existir um produto onde a única informação disponível é: Vinho reserva. A probabilidade do produto ser identificado incorretamente na BD-B será muito elevada pois, a informação disponível permite identificar este produto em qualquer cor de vinho que pertença a uma reserva. Utilizando um modelo ontológico, onde o indevido vinho fosse portador de toda a informação que o caracteriza, seria possível inferir que sendo um vinho e não tendo cor seria considerada a cor tinto (por omissão). Automaticamente o vinho iria herdar todas características desse indevido.

Este processo é muito demorado e pesado pois cada indevido tem de ser descrito assim como as condições que definem as características mínimas conhecidas para identificar a quem pertence o indevido. Em contrapartida com um modelo ontológico bem construído é possível obter um fiabilidade e exatidão extraordinária na integração de sistemas heterogéneos.

Sendo um processo tão demorado e complexo, não era humanamente possível fazer a criar todo o modelo ontológico, optamos por nos focar na família das bebidas alcoólicas, mais especificamente vinhos.

4.4 Pesos

Para poder fazer a correlação entre dois sistemas heterógenos é necessário existir um mecanismo de pesos. É sabido que não existe a possibilidade de fazer uma relação através de um código, chave primária, nem através de uma comparação de strings. Daí a necessidade de criar um sistema de semântica. Sabendo que nos processos anteriores os produtos foram descritos e foi possível obter mais informação é essencial atribuir pesos às diferentes características pois nem todas têm a mesma relevância. Por exemplo, no produto: Vinho tinto fermentado. O peso do vinho seria o mais elevado pois representa uma família/classe, já entre as duas propriedades seria o tinto que teria um peso maior. Isto para destinar entre vinho branco, verde, sendo que ambos também resultam de um processo de fermentação.

Existem várias maneira de implementar um mecanismo de pesos, sendo que neste caso optou-se por criar um ficheiro em xml só com o valor dos pesos e a propriedade/classe a que este se refere. Está solução é extremamente útil e prática pois permite modificar o valor dos pesos rapidamente. O valor está localizado num único ficheiro. Ao mesmo tempo também é fácil aceder ao valor dos mesmos utilizados um parser *XPath* pois, os produtos estão descritos em xml e o ficheiro dos pesos respeita esses *name spaces.*

A figura 4 é apenas um pequeno excerto dos pesos.

```
<xml version="1.0" encoding="utf-8">
  <classe familia="bebidas">80</classe>
  <propriedade atributo="geral">5</propriedade>
  <propriedade atributo="grandeza">20</propriedade>
  <propriedade atributo="quantidade">20</propriedade>
  <propriedade atributo="cor">40</propriedade>
  <propriedade atributo="ano">5</propriedade>
  <propriedade atributo="origem">10</propriedade>
  <propriedade atributo="tipoprocessamento">20</propriedade>
```

Figure 4 - Extracto dos Pesos

4.5 Matching dos Produtos

Este é o processo final, e o mais palpável pois, é daqui que sai o ficheiro final com relação do produto entre ambas as base de dados. Ou seja, para um produto X da BD-A é encontrado o seu correspondente em BD-B e as descrições do produto em ambos os sistemas heterogéneos são combinadas de forma a fazer uma única entrada no ficheiro

de output. A figura 5 é o exemplo do ficheiro de output para o produto "Vinho botba 0.75 cl tinto" presenta na BD-A.

```
- <xml version="1.0" encoding="utf-8">
  - <linha valor="vinho borba 0.75 cl tinto">
      <produto familia="bebida">vinho</produto>
      <propriedade atributo="marca">borba</propriedade>
      <propriedade atributo="quantidade">0.75</propriedade>
      <propriedade atributo="grandeza">cl</propriedade>
      <propriedade atributo="cor">tinto</propriedade>
      <propriedade atributo="tipobebida">alcoolicas</propriedade>
      <propriedade atributo="tipoprocessamento">fermentadas</propriedade>
      <propriedade atributo="classificacao">maduro</propriedade>
    </linha>
    ...
</xml>
```

Figure 5 - Exemplo do resultado final de *matching* de um produto

Para atingir este resultado existe uma sequência de tarefas que são executadas através do projeto *SWEquals,* usando a linguagem Java. O ponto de partida é os ficheiros resultantes dos processos anteriores, sendo eles o BD-A_source.xml, BD-B_source.xml, WSFood.owl e pesos.xml. Os dois primeiros são o resultado do processamento gramatical e lexical. O segundo é o modelo ontológico e por fim temos o ficheiro com o valor dos pesos. Este projeto vai pegar nestes ficheiros para realizar as seguintes tarefas:

1. Fazer *parse* do BD-A_source.xml e BD-B_source.xml para os POJOs com recurso a XPath;
2. Para cala produto, executa o modelo ontológico de forma a inferir características desconhecidas (na prática é adicionar essas características à lista de propriedades do objecto Produto);
3. Após todos os produtos se encontrarem em memória vai procurar encontrar o produto da BD-B com maior score. Para isso apenas irá somar os pesos das características comuns em ambos os *data sources*. Por outras palavras, para um produto em BD-A vai percorrer todos os produtos da BD-B. Em cada produto da BD-B é calculado o somatório total dos pesos das características que são comuns aos dois. No final obtêm-se o produto em BD-B com maior score.
4. Por fim, cada produto *matched* é gravado no ficheiro de output com ambas as características presentes na BD-A e BD-B, evitando repetir as mesmas (figura 5).

A figura 6 representa um esquema geral do processo de mapeamento dos produtos entre os diferentes sistemas heterógenos.

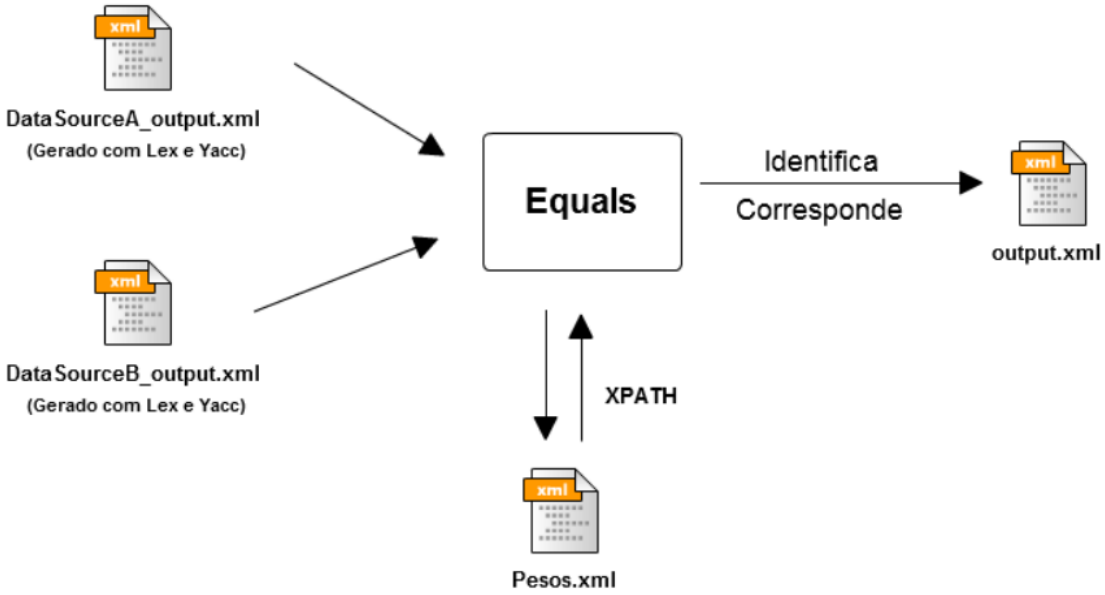

Figure 6 - Esquema geral do processo de *matching*

5 FUNCIONAMENTO DA APLICAÇÃO

O funcionamento da aplicação é bastante simples e prático. Para executar a aplicação basta abrir uma consola pasta da aplicação e executar o *script* "run.sh". Este *script* está escrito em *bash* e é responsável por executar todos os paços do workflow do sistema. Desde executar as os programas em Java passado pela compilação e execução do Flex e Bison. No final, o resultado pode ser consultado no ficheiro output.xml, na raiz do projeto. Como o próprio nome indica, o ficheiro apresenta os dados em formato de XML. A única configuração que pode ser necessária, e opcional, é os dados dos sistemas heterogéneos terem de estar na raiz do projeto com os seguintes nomes: BD-A_source.txt e BD-B_source.txt. A figura 7 representa o exemplo do *output* para um produto da BD-A com BD-B.

```
- <xml version="1.0" encoding="utf-8">
  – <linha valor="vinho borba 0.75 cl tinto">
      <produto familia="bebida">vinho</produto>
      <propriedade atributo="marca">borba</propriedade>
      <propriedade atributo="quantidade">0.75</propriedade>
      <propriedade atributo="grandeza">cl</propriedade>
      <propriedade atributo="cor">tinto</propriedade>
      <propriedade atributo="tipobebida">alcoolicas</propriedade>
      <propriedade atributo="tipoprocessamento">fermentadas</propriedade>
      <propriedade atributo="classificacao">maduro</propriedade>
    </linha>
    ...
</xml>
```

Figure 7 - Exemplo de um produto

6 Conclusões e Trabalho Futuros

Este trabalho pretendeu demostrar a viabilidade e fiabilidade em utilizar integração semântica para resolver problemas de integração de sistemas heterogéneos. A pesar do peso e custo da construção de sistemas deste tipo ser considerável, pode-se concluir que esta é a solução mais viável. Nem sempre existe a possibilidade centralizar a informação numa fonte de dados partilhada, sendo porque o sistema já está em produção ou porque a detentora dos dados não tem vantagens em alterar o seu sistema. Para além disto, estes sistemas permitem descrever os dados de forma a que tenham significado para a máquina, descrevendo linguagens através de sintaxes.

De momento o sistema encontra-se a funcionar para um leque de produtos reduzido, bebidas. Isto deve-se em parte à grande complexidade da modelação dos dados com em conjunto com a nossa inexperiência neste tipo de problemas e tecnologias. Isto para não referir o número de produtos e características das duas fontes de dados. Assim sendo no futuro pretende-se desenvolver e melhorar as seguintes tarefas:

- Completar o modelo ontológico;
- Completar a analise lexical e gramatical;
- Fazer stemming ao vocábulos;
- Fazer melhoramento dos pesos através de ferramentas de treino automáticas (JMetal).

7 REFERÊNCIAS

[1] Flex [Online] http://flex.sourceforge.net/

[2] Bison [Online] http://www.gnu.org/software/bison/

[3] Yacc [Online] http://dinosaur.compilertools.net/yacc/index.html

[4] Protégé [Online] http://protege.stanford.edu/

[5] Owlapi [Online] http://owlapi.sourceforge.net/documentation.html

[6] HermiT [Online] http://hermit-reasoner.com/