PRC TPF

António Chaves
A75870

June 2019

1 Introdução

A presente secção tem em vista a consolidação de todo o processo de desenvolvimento e utilização da aplicação desenvolvida no âmbito da unidade curricular de Processamento e Representação de Conhecimento, no ano letivo de 2018/2019. O enunciado do trabalho propôs o desenvolvimento de uma aplicação WEB capaz de explorar toda a informação proveniente de um dataset que, depois de convertido para a extensão .ttl, fosse incluido na instância local de um servidor GraphDB. Vinícios é o resultado final da submissão.

2 Instalação

Apenas é necessário, dentro das diretorias vinicios-client/ e server/ correr o comando npm install. A BD incluida no GraphDB deve ser chamada PRC-TP para que a sua leitura seja correta e importado o ficheiro output.ttl da diretoria scripts/.

3 Dataset

O contexto da aplicação e todos os dados nela presentados são provenientes do dataset Wine Reviews que contém a informação relativa a reviews de cerca de 130 mil vinhos de todo o planeta. Os campos incluem a generalidade da descrição dos vinhos (preço, castas, designação), do enólogo que o descreveu e pontuou, Cooperativa e país de proveniência.

4 Ontologia

A Figura 1 resume o resultado final da Ontologia, omitindo os campos de data properties. As classes são originadas do estudo do dataset utilizado, ligeiramente adequadas às necessidades da aplicação.

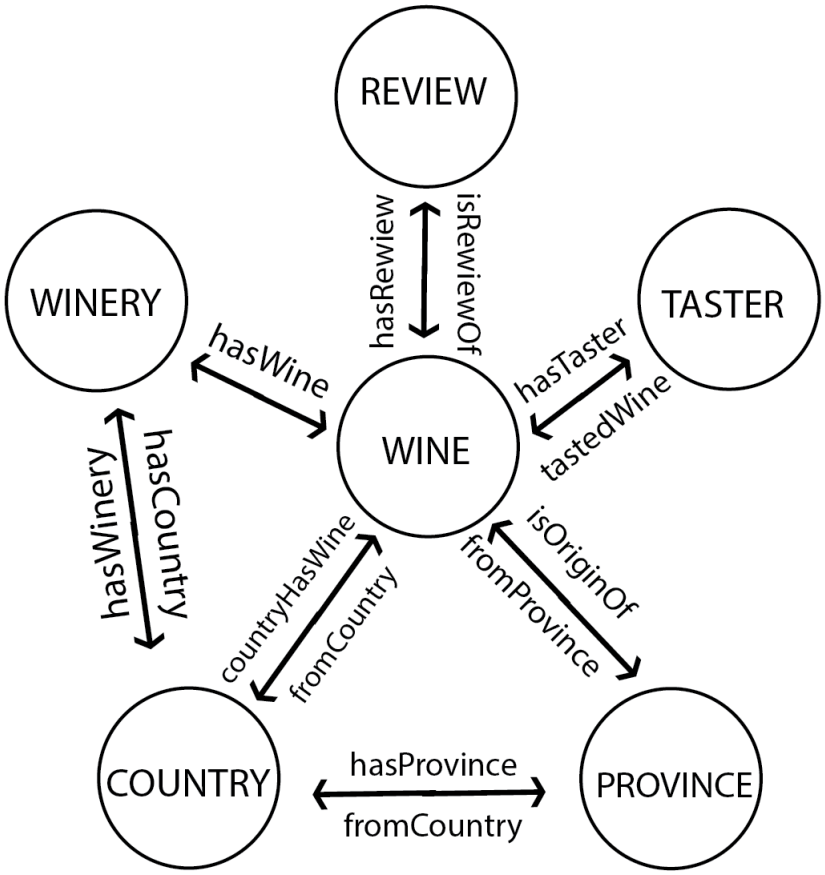

Figure 1: Esquema de classes e object properties da Ontologia

5 Conversão do Dataset

Dada a extensão do dataset obtido, foi necessária a sua conversão para um formato compatível com o GraphDB, neste caso, .ttl, seguindo os seguintes passos:

- Tratamento de caracteres especiais: os identificadores gerados apenas podem conter caracteres alfanuméricos e uma seleção de símbolos. Assim, com ajuda do módulo re, foram substituídos todos os caracteres indevidos, para nào causar conflito na importação do ficheiro final.
- Leitura das entradas .JSON: o ficheiro é fornecido automaticamente, desde que se encontre na pasta de execução do script (alternativamente, através da alteração do caminho pré-definido) e, para cada entrada, são preenchidos até 6 dicionários, cada um correspondende a cada uma das classes da Ontologia. Isto assegura que, aquando da conversão final, todas as Object Properties de cada indivíduo serão incluídas.
- Geração de ID de países: de modo a fazer uso de uma API externa, foi adicionado um ID a cada país, de acordo com o protocolo em Country Flags .io . Posteriormente, este ID será utilizado para fazer corresponder a cada país a sua bandeira nacional. Conversão de dicionários: os dicionários de cada classe são convertidos em formato textual para a linguagem Turtle e concatenados com o ficheiro gerado pelo Protégé, que serve como cabeçalho da Ontologia. O número de entradas lidas foi limitado devido ao consumo energético e temporal do número original de entradas.
- Tratamento de caracteres especiais: os identificadores gerados apenas podem conter caracteres alfanuméricos e uma seleção de símbolos. Assim, com ajuda do módulo re, foram substituídos todos os caracteres indevidos, para nào causar conflito na importação do ficheiro final.
- Leitura das entradas .JSON: o ficheiro é fornecido automaticamente, desde que se encontre na pasta de execução do script (alternativamente, através da alteração do caminho pré-definido) e, para cada entrada, são preenchidos até 6 dicionários, cada um correspondende a cada uma das classes da Ontologia. Isto assegura que, aquando da conversão final, todas as Object Properties de cada indivíduo serão incluídas.
- Geração de ID de países: de modo a fazer uso de uma API externa, foi adicionado um ID a cada país, de acordo com o protocolo em Country Flags .io . Posteriormente, este ID será utilizado para fazer corresponder a cada país a sua bandeira nacional. Conversão de dicionários: os dicionários de cada classe são convertidos em formato textual para a linguagem Turtle e concatenados com o ficheiro gerado pelo Protégé, que serve como cabeçalho da Ontologia. O número de entradas lidas foi limitado devido ao consumo energético e temporal do número original de entradas.

A importação para o GraphDB gerou cerca de 30 mil statements, dos quais, 25 mil são explícitos.

6 Backend

Fazendo recurso da linguagem NodeJS, foi desenhado um servidor cuja funcionalidade é responder a todos os pedidos provenientes do Frontend da aplicação. As suas rotas são muito específicas e respondem apenas com informação JSON proveniente do envio de queries SPARQL.

7 Frontend

Por fim, a aplicação criada em VueJS. O objetivo maior do Frontend da aplicação (a par com a conexão com o seu contexto) foi a inclusão de componentes diversos, de modo a explorar um pouco mais as possibilidades do desenvolvimento em Vue. Em vários componentes é visível o uso de componentes de Grelha em funcionamento com diferentes breakpoints da página, para que a sua visualização não seja afetada por diferentes tamanhos de ecrã. Os componentes como WineListing ou SelectedWine dependem da correta utilização das rotas to backend e calculam as suas estruturas de dados aquando da criação do DOM da página. Inicialmente, é enviado um pedido get para receber os dados relativos ao componente, com a finalidade de os dispor em elementos ordenados (Tabelas, grelhas,...). Durante todo o processo de desenvolvimento houve um foco especial na estilização dos componentes, de modo a tentar tornar a aplicação o mais próxima possível de uma real iteração do problema. Foi com o mesmo intuito que foi desenvolvido o logotipo, baseado em elementos alusivos ao contexto, como a garrafa de vinho e o copo de degustação.