Scripting no Processamento de Linguagem Natural Trabalho Prático 3

Etienne Costa - A76089 Gonçalo Pinto - A83732
Luís Ribeiro - A85954

(25 de junho de 2021)

Resumo

Para vários projetos de software, é habitual soluções envolvendo vários ficheiros, várias pastas. O presente relatório descreve o processo de implementação de análise, de planeamento e de implementação de *scripts* em *Python* que permite criar um *Template multi-file* e também um *Template multi-directory* em que ambos pretendem auxiliar o processo de criação de projetos de software. Deste modo, desenvolveu-se um sistema que permitisse criar uma estrutura de diretorias e ficheiros baseando num *template* e um outro sistema que permitisse criar um *template* com base numa determinada diretoria.

1 Introdução

No 2º semestre do 1º ano do Mestrado em Engenharia Informática da Universidade do Minho, existe uma unidade curricular enquadrada no perfil de Processamento de Linguagens e Conhecimento denominada por Scripting no Processamento de Linguagem Natural, que tem como objetivo agilizar a capacidade dos alunos no processo de automatização de uma série de tarefas ligadas a ficheiros, textos e informação em linguagem natural, desenvolver a aptidão em lidar com grandes quantidades de informação e potenciar a aprendizagem de uma programação documental heterogénea.

O presente trabalho enquadra-se nesta unidade curricular e pretendeu-se realizar um trabalho de análise, de planeamento e de implementação de *scripts* em *Python* que permitisse criar um *Template multi-file* e um *Template multi-directory*.

2 Contextualização

Para vários projetos de software, é habitual soluções envolvendo vários ficheiros, várias pastas. Exemplo: um ficheiro, uma *makefile*, um manual, uma pasta de exemplos, etc.

O *script "mkfromTemplate"* é capaz de aceitar um nome de projeto e um ficheiro, um *template-multi-file*, criando os ficheiros e pastas iniciais do projeto.

O *"mkfromDirectory"* efetua o processo inverso ao do *"mkfromTemplate"*, isto é, é capaz de aceitar uma diretoria e gera um ficheiro *template-multi-file* idêntico ao que o *"mkfromTemplate"* aceita. Esta diretoria inclui além de subdiretorias e ficheiros, um ficheiro designado de *meta* com os metadados.

3 Conceitos

Como foi apresentado anteriormente desenvolveu-se um *Template multi-file* e um *Template multi-directory* de forma a dar apoio no processo de criação de projetos. Para isso, foi necessário identificar alguns conceitos importantes.

3.1 Metadados

Um desses conceitos nestes sistemas foi o conceito de metadados que são informações, como por exemplo, o autor do projeto ou o email. Os metadados facilitam o entendimento dos relacionamentos e a utilidade das informações dos dados. No *script "mkfromTemplate"* esta informação é substituída nos conceitos seguintes, já no *script "mkfromDirectory"* esta informação encontra-se num ficheiro específico que posteriormente é integrada no *template* gerado.

3.2 Tree

Como foi referido estes *scripts* envolvem vários ficheiros e várias pastas logo é essencial descrever esses mesmos. Para tal descrição decidiu-se utilizar estrutura de diretorias e ficheiros que no caso do *script "mkfromTemplate"* serão criadas e no *script "mkfromDirectory"* serão percorridas de forma a escrever no *template* a estrutura da diretoria e o conteúdo dos ficheiros que a compõem.

3.3 Template

Tal como o conceito anterior, este conceito também é essencial para os *scripts* criados, pois é utilizado por ambos. Um *template* inclui metadados, uma árvore de diretorias e/ou ficheiros e o *template* de cada ficheiro. O metadado *"name"* vai ser processado via argumento de linha de comando. No *script "mkfromTemplate"* o objetivo é a partir de um *template* criar uma *tree*, no *script "mkfromDirectory"* é a partir de uma *tree* criar um *template*.

4 A Implementação

Nesta secção iremos discutir a implementação dos dois sistemas implementados. Ambos foram desenvolvidos na linguagem de programação, *Python*, pois possui uma sintaxe simplificada e não complicada, o que dá mais ênfase à linguagem natural. Devido à sua facilidade de aprendizagem e uso, o código pode ser facilmente escrito e executado muito mais rápido do que em outras linguagens de programação. Outra das vantagens desta linguagem é quantidade elevada de módulos disponíveis que auxiliam na construção dos sistemas desenvolvidos. Neste projeto utilizou-se os seguintes módulos:

- *os* - fornece métodos de usar as funcionalidades do sistema operacional;
- *re* - fornece operações de correspondência através de expressões regulares;
- *sys* - fornece o acesso a variáveis usadas ou mantidas pelo interpretador;
- *yaml* - fornece um analisador da linguagem YAML;
- *jjcli* - fornece a simplificação da criação de filtros Unix;

4.1 mkfromTemplate

Como foi referido efectuou-se um *script* capaz de aceitar um nome de projeto e *template*, criando assim os ficheiros e pastas iniciais do projeto. Para isso o sistema efetua as seguintes etapas:

1. Leitura do conteúdo presente no template introduzido;
2. Substituição da *tag* {*%name%*} pelo nome introduzido como parâmetro no conteúdo lido;
3. Considerando que o template possui cada conceito incializado com a *tag* ===, efetou-se uma separação por esta, obtendo uma lista de conteúdos onde o primeiro elemento da lista é referente aos metadados, o segundo à *tree* e os restantes elementos ao *template* de cada ficheiro;
4. Consequentemente efetuou-se a extração da informação presente nos metadados, para isso percorreu-se cada linha do conteúdo referente a *tag* meta e considerou-se que os dados encontram-se separados por *":"* onde o elemento à esquerda deste corresponde à chave e à direita o valor. Assim, construiu-se um dicionário de forma a poder substituir as chaves pelos valores respetivos nos ficheiros a criar;
5. O próximo passo deste algoritmo passou por percorrer o dicionário acabado de criar onde se substitui na lista de conteúdos as chaves pelos valores respetivos;

6. Criado esta estrutura realizou-se a construção de uma lista onde cada elemento vai corresponder ao *path* de uma determinada diretoria ou ficheiro presente no conteúdo da *tag tree*. Este conteúdo encontra-se organizado por níveis através do símbolo -", isto é, a quantidade de símbolos representa a profundidade na diretoria, para isso, cada diretoria tem de ser terminada por "/". Portanto para a construção da lista é percorrido o conteúdo da *tree* onde para cada elemento se verificou o nível, o conteúdo e se é uma diretoria tendo sempre em consideração o nível anterior no sentido de verificar se a diferença for 0 é um elemento ao nível da pasta atual e caso seja uma diretoria é aumentada a profundidade atual. Por outro lado, a diferença pode ser de 1 nesse caso é uma ficheiro ou diretoria dentro da pasta anterior representada. Um outro cenário possível pode acontecer, quando a profundidade atual é mais elevada do que a profundidade indicada no template, nesse caso é necessário aceder ao caminho desse nível para isso todo este processo é auxiliado por um dicionário onde nas chaves se guardou o nível e no valor o *path* da diretoria correspondente;

7. Por fim, o último passo deste *script* diz respeito à criação das diretorias, bem como, dos ficheiros e sua escrita. Para isso, é percorrida a lista de caminhos construído do ponto anterior onde se verifica se o elemento que faz parte desta é uma diretoria, se assim for é criado a diretoria, caso contrário, é criado e aberto o ficheiro onde se procura na lista de conteúdo lida, o nome deste ficheiro e posteriormente escreve-se a informação presente no *template* .

4.2 mkfromDirectory

Este *script* é capaz de aceitar uma determinada diretoria com subdiretorias e ficheiros, gerando um ficheiro *template* idêntico ao que o *"mkfromTemplate"* aceita. Assim sendo, o sistema efetua as seguintes etapas:

1. A primeira etapa deste *script* é verificar se o utilizador inseriu uma diretoria, para isso recorreu-se à função disponível *os.path.isdir* do módulo *os* que verifica se o argumento fornecido é ou não uma diretoria;

2. Caso se verifique, é construído a árvore de diretorias e ficheiros, partindo da diretoria fornecida com o método *os.walk*. A partir daí, são percorridas todas as diretorias de cada nível, guardando em *string* a informação relativa ao nível e ao conteúdo associado, de modo apresentar a *tree* da diretoria respeitando o formato *template* esperado;

3. Posteriormente é verificado se na diretoria fornecida existe o ficheiro dos metadados denominado *meta*. Se existir, com auxílio do módulo *yaml*, é carregado para uma variável a informação desse ficheiro;

4. Para terminar, a última etapa é construir o ficheiro *template* partindo da informação recolhida previamente. Desse modo é criado o ficheiro, de seguida são escritos os metadados (chaves e valores) previamente marcados com a *tag "=== meta"*, posteriormente é escrita a *tag "=== tree"* com o formato definido no ponto 2 e finalmente são percorridos todos os items do dicionário de ficheiros onde é escrito o seu nome como *tag* e o conteúdo que possui.

5 Conclusão

O presente relatório descreveu, de forma sucinta, o trabalho de análise, planeamento e de implementação de *scripts* em *Python* que permitisse a manipulação de vários ficheiros e várias pastas.

Durante a elaboração deste trabalho surgiram algumas dificuldades que, com o esforço e entusiasmo do grupo pelo resultado, acabaram por ser ultrapassadas. Entre as quais destacam-se o desafio que foi efetuar o *parsing* do *template* no *script "mkfromTemplate"*.

Numa perspetiva futura, considera-se que seria interessante a expansão do *script "mkfromTemplate"* para algo mais completo e robusto, isto é, em vez de se escrever diretamente a informação de cada ficheiro no *template* poder-se-ia previamente verificar se esse conteúdo é válido mediante o tipo de ficheiro criado.

Após a realização deste trabalho, o grupo ficou consciente da importância de automatização de uma série de tarefas no processamento de grandes quantidades de informação. Consideramos que os objetivos propostos com a realização deste trabalho foram cumpridos, bem como, a consolidação dos conhecimentos em *scripting* no processamento de linguagem natural.

Por fim, o grupo espera que os conhecimentos obtidos e consolidados sejam de enorme utilidade tendo em conta uma perspetiva futura.