Relatório

Processamento de Linguagens

Alunos
José Cosgrove 18826
André Cardoso 18848

Professor
Alberto Simões

Licenciatura em Engenharia de Sistemas Informáticos

Barcelos, 22 de novembro de 2020

Resumo

Foi proposto para trabalho prático desta Unidade Curricular a elaboração de um projeto em Python, onde puséssemos em prática conceitos adquiridos nas aulas. Utilizamos ferramentas como o Lexer do Ply (ply.lex) e o Regex (re) para auxiliar na criação de código. Utilizamos ainda HTML para fazer o output dos dados.

Índice

1. TAP – Test Anything Protocol

Este é um protocolo de impressão de resultados de testes executados a unidades de software.

Segue um formato constante de identificação de testes: estado, posição e descrição. Os subtestes estão formatados do mesmo modo. Para além de testes e subtestes, pode encontrar-se uma linha identificadora do total de testes ou sequência de subtestes – esta encontra-se no fim ou início da sequência. Também é possível encontrar-se comentários, linhas de texto não relevantes ao processamento.

Todo o seu conteúdo de um ficheiro TAP tem de ser reconhecido para uma boa execução do programa criado, pois caso alguma expressão não seja reconhecida, todo o processo falha.

2. REGEX – Expressões Regulares

De modo a ler os ficheiros e reconhecer devidamente os elementos inseridos, fazemos uso de expressões regulares, a ferramenta Regex.

Uma expressão regular é uma sequência de caracteres que definem um padrão, neste caso, o padrão com que os ficheiros TAP são escritos.

- **Total**

Toda a sequência de testes ou subtestes tem, ou no início ou no fim desta, uma indicação do seu total, por exemplo:

1..20

Visto que toda a sequência lógica de testes e subtestes deverá iniciar pelo primeiro ou seja o número '1', podemos considerar o primeiro valor constante, assim como os pontos entre os dois números. Quanto ao número referente ao total, podemos assumir que pelo menos um número entre '1' e '9' tem de ser e pode ou não ser um número com mais de dois dígitos.

Sabendo isto podemos deduzir que a expressão regular que define o total de casos da sequência é:

"1\.\.[1-9]\d*"

- '1'

  Caracter '1'

- '\.'

  Dado que o caracter '.' representa um caracter especial no regex, este para ser interpretado literalmente tem de ser acompanhado por '\'.

- '[1-9]'

  Quando colocámos caracteres entre '[' e ']' indicamos que deve ser escolhido apenas um deles. No entanto, se colocarmos o '–' entre dois caracteres, é identificada uma sequência ASCII e assim é escolhido um valor dentro dessa sequência, neste caso entre '1' e '9'.

- '\d*'

  Este caracter representa qualquer dígito, é o equivalente a "[0-9]". Quando acompanhado por '*', este indica que pode repetir o reconhecimento de qualquer dígito zero ou infinitas vezes. O caracter '*' é aplicável a qualquer expressão válida.

- **Estado**

Varia apenas entre “ok” e “not ok” e indica o sucesso da avaliação. Resume-se à seguinte expressão:

“ok|not ok”

Dado que os únicos valores possíveis para este campo são conhecidos e não variáveis, aplicamos uma expressão definida. O caracter ‘|’ representa a operação lógica OR, permitindo reconhecer uma das duas expressões intercaladas.

- **Posição (Offset)**

A posição relativa dos testes e subtestes varia do mesmo modo que o valor total de testes e subtestes ([1-9]\d*), assim sendo:

“\ ([1-9]\d)(\ |\n)”*

O lexer do ply oferece-nos a possibilidade de criar um token com os caracteres que desejamos ignorar sendo que o utilizamos várias vezes nas aulas para ignorar espaços e quebras de linha. No entanto, neste trabalho utilizar essa ferramenta desse modo revelou-se um desafio pois sempre que um teste não apresentava uma descrição imediatamente a seguir – portanto, avançava para o próximo teste com uma quebra de linha – o lexer tinha problemas em identificar os tokens.

Assim, decidimos identificar ‘manualmente’ os espaços e quebras de linha.

- *‘(\ |\n)’*

A operação lógica causa a escolha entre os caracteres ‘\ ’ e ‘\n’ – o primeiro quando encontra Texto ou Comentário de seguida, o segundo quando avança diretamente para o próximo teste.

- **Texto**

Cada teste pode apresentar uma pequena descrição. Esta pode conter palavras ou números, assim é reconhecível deste modo:

“-(\ [\w\d])*\n”*

Uma descrição pode conter uma sequência de palavras ou dígitos ou até um misto dos dois. No entanto, vai sempre começar com um ‘–’. Para além disso, as palavras terão sempre a antecedê-las um caracter espaço (‘\ ’).

- ‘\w’

Este caracter inclui todos os caracteres que se podem encontrar a constituir uma palavra.

- **Comentário**

Tal como o Texto, podemos encontrar este elemento imediatamente a seguir à Posição do teste:

"\#((.))*\n"*

No entanto, tendo em conta que estes comentários não são regulados, a sua informação pode não ser relevante, sendo assim ignorado.

- **Indentação**

Presente nos subtestes, é a nossa distinção de níveis de profundidade destes. Deparamo-nos com um problema aqui, pois este entrava em conflito com a variável 't_ignore' previamente existente:

"(\t)|(\ \ \ \ )"

A variável impedia a identificação das indentações realizadas através de espaços, assim vimo-nos forçados a remover esse espaço e a complementar os tokens que necessitassem.

3. Formatação do Ficheiro

Estando nós a trabalhar protocolos TAP, temos de preparar o programa para todas as eventualidades de formatação, tal como as indentações devido à existência de subtestes.

Um dos problemas com que nos deparamos foi a indentação dos subtestes e o seu conflito com uma funcionalidade anteriormente utilizada. Apesar de esse problema ser facilmente solucionado, o armazenamento e identificação dos subtestes continua presente. A identificação dos subtestes é idêntica à dos testes à exceção da profundidade – não existente no caso dos testes. Foi facilmente resolvido com a utilização de um contador que incrementa a cada Indentação identificada e anulada sempre que os dados são armazenados ou é identificado um comentário sem informação referente a subtestes antes.

Para além destes casos, a formatação dos ficheiros não se revelou um problema muito intensivo.

4. PLY – LEXER

É esta ferramenta que torna todo o trabalho possível, devido à possibilidade de identificar os tokens previamente numa variável, e de seguida identificar independentemente as condições em regex identificadoras de cada elemento.

Mais do que isso, após ceder o input, é através deste que somos capazes de selecionar cada elemento existente – utilizando os identificadores previamente criados associados pelo nome à lista de tokens – e processá-lo do modo correto.

5. Implementação HTML

Uma vez que já temos acesso aos dados e sua estrutura temos de os apresentar.

Poderíamos apresentar os dados na consola, mas isso não iria ter a apresentação e interpretação que desejávamos. Uma boa solução para este problema era a utilização dos dados através de uma página HTML.

Esta página é criada utilizando a framework bootstrap que produz um aspeto visual mais apelativo e ainda várias classes e formatações já definidas.

É inicialmente criado um ficheiro através dos métodos de ficheiros de python e é escrito nesse ficheiro as tags do html e são inseridos através do python, no ficheiro HTML, os dados de leitura.

A página começa por apresentar os diversos ficheiros que são lidos. Cada título de ficheiro apresenta-se a verde ou vermelho, para representar se os testes correram todos bem ou se houve erros.

Cada ficheiro tem um número de testes, a cada teste podem corresponder subtestes, a cada subteste podem corresponder outros subtestes e assim sucessivamente.

Estes testes e subtestes são também apresentados com a indentação correspondente, com cor verde, caso esteja tudo ok, e a vermelho se não estiver.

6. Conclusão

Na elaboração deste projeto, pudemos aprofundar mais os nossos conhecimentos no que toca a trabalhar em python (era uma linguagem nova até ao início do semestre) e tentar pôr em prática todas as boas práticas que esta linguagem implica.

Tivemos também de nos colocar à prova em termos criativos e lógicos, pois existiam diferentes formas de pensar e elaborar este projeto.

No que toca ao trabalho em equipa, conseguimos fazer uma correta gestão e partição do trabalho e fomo-nos ajudando um ao outro sempre que algum de nós se encontrava com mais dificuldades.

Achamos o trabalho algo acessível, uma vez que sempre fomos a todas as aulas, o que ajudou bastante no desenvolvimento do projeto, pois ao longo das aulas fomos fazendo casos práticos que envolveram conceitos que utilizamos neste projeto.

Para finalizar o relatório, apenas acrescento a ideia de que achamos este tipo de trabalhos muito importantes no nosso percurso académico, pois achamos que eles nos vão preparando cada vez mais para sairmos deste curso engenheiros capazes e capacitados.