Universidade do Minho

Mestrado Integrado em Engenharia Informática

Gramáticas na Compreensão de Software

Q&A do FAQ da Univerdade do Minho

a78679 - Diana Ribeiro Barbosa
a78821 - José Carlos Lima Martins

7 de Janeiro de 2019

Resumo

Neste trabalho é relatado o desenvolvimento de um Q&A System com área de conhecimento a FAQ da Universidade do Minho. De forma a desenvolver este sistema, foi usado os conhecimentos de programação de linguagens. Dentro desses conhecimentos, foi aplicado as Gramáticas Independentes de Contexto e Gramáticas de Atributos, usando a notação ANTLR.

Conteúdo

1 Introdução

O objetivo deste trabalho é o desenvolvimento de um *Q&A System*, isto é, um sistema de perguntas e respostas aplicando os conhecimentos adquiridos ao longo do semestre na unidade curricular de *Gramáticas na Compreensão de Software* sobre gramáticas de atributos, gramáticas independentes de contexto, qualidade de gramáticas, entre outros.

Um *Q&A System* [2] é um programa que tenta responder a perguntas colocadas por utilizadores. Como exemplos de *Q&A Systems* pode-se tomar o *Quora*, o *Yahoo! Answers* e o *Stack Overflow*. Normalmente, cada um destes sistemas tem uma área de conhecimento na qual opera, possuindo as respostas às principais perguntas relativas a área. A área de conhecimento que escolhemos é a *FAQ* da Universidade do Minho.

Assim, na secção 2 damos a conhecer e aprofundamos um pouco sobre a área de conhecimento escolhida. Na secção 3, são indicadas as funcionalidades suportadas pelo sistema desenvolvido. Numa outra secção, 4, é explicada a GIC construída e a estrutura lógica do programa. Além disso, na secção 5, são apresentadas as funções e uso de atributos que suportam as funcionalidades disponibilizadas. Por fim, na secção 6, é feita uma análise crítica ao trabalho e possíveis melhorias.

2 Área de Conhecimento

A área de conhecimento escolhida para o trabalho desenvolvido foi o *FAQ (Frequently Asked Questions)* [1] da Universidade do Minho. O seu objetivo é responder às

questões mais frequentes colocadas pelos alunos diminuindo o volume de emails enviados e de idas aos vários serviços da universidade (e.g. Serviços Académicos, Serviços de Ação Social, Serviços de Documentação, Serviços de Relações Internacionais, Direção de Recursos Humanos, ...). Essencialmente aumentar a sua eficiência. Assim, tem como vantagem a redução do tempo de espera quer de resposta aos *emails* quer das filas no atendimento presencial.

O tema da informação guardada na base de conhecimento é a Universidade do Minho, nomeadamente, informação relativa ao calendário escolar, pagamento de propinas, admissão aos cursos, recrutamento de investigadores, residências universitárias, ECTSs, inscrições, campi e matrículas, entre outros. Quanto aos utilizadores, o sistema destina-se a uso por parte de alunos (que já frequentem a universidade ou que o pretendam fazer no futuro), *alumni*, corpo docente, investigadores, funcionários ou membros da direção.

Alguns exemplos de perguntas possíveis a que o *Q&A System* poderá dar resposta (contando que a informação seja corretamente inserida na base de conhecimento) são:

- O que são ECTSs?
- Onde me posso candidatar à Universidade do Minho?
- Como posso fazer *Erasmus*?
- Onde me devo dirigir para assuntos relacionados com bolsas de estudo?
- Qual é o calendário escolar da Escola de Engenharia?

3 Funcionalidades suportadas

O *Q&A system* desenvolvido suporta várias funcionalidades que consideramos essenciais. O nosso objetivo era a criação de um sistema eficiente e intuitivo. Neste sentido, delineamos um conjunto de metas a atingir ao longo da realização do projeto, nomeadamente, funcionalidades a serem suportadas pelo sistema a nível da sua capacidade de resposta.

A lista é a que se segue:

- *Case* das letras (maiúsculo/minúsculo) não interferir com o reconhecimento da pergunta;
- Reconhecer perguntas que terminem em vários/múltiplos sinais de pontuação;
- Devolver a resposta guardada na Base de Conhecimento cuja intenção seja o mais semelhante à questão colocada;
- Caso não haja uma correspondência que se destaque de entre as existentes, havendo várias com o mesmo nível de igualdade, devolver essas mesmas respostas;
- Informar o utilizador da ausência de informação relativa à questão colocada;
- Reconhecer várias perguntas numa só questão.

Estes objetivos foram alcançados, sendo possível verificar na figura 1 *inputs* sintaticamente corretos, isto é, que o sistema reconhece conforme as regras definidas.

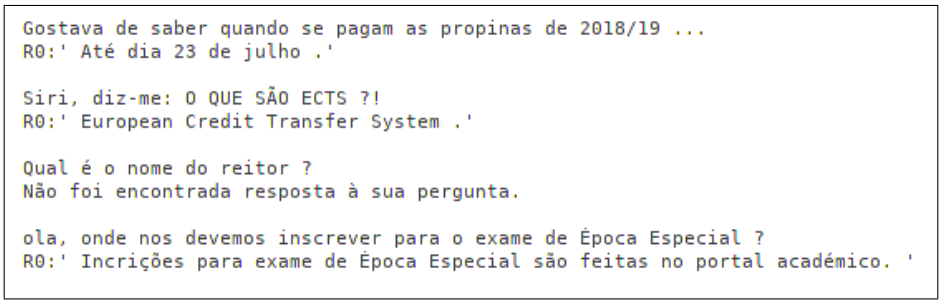

Figura 1: Exemplo do funcionamento do *Q&A System FAQ UMinho*

4 Gramática Independente de Contexto

É possível identificar duas componentes principais do sistema: a Base de Conhecimento - composta por pares (pergunta,resposta) - e as Questões colocadas pelos utilizadores. Na prática, estas correspondem às seguintes produções definidas na gramática:

'bcQAS': define a estrutura da base de conhecimento, nomeadamente, pares (intenção, resposta). A intenção é o equivalente à pergunta. Nela estão guardados o tipo (e.g. 'Qual', 'Como', 'Onde'), a ação (e.g.'imprimir', 'inscrever') e as *keywords* associadas (e.g. 'propinas', 'regulamento', 'exame'). Este conhecimento é lido e guardado para possibilitar a resposta às questões. Caso se deseje expandir a base de conhecimento e os novos pares incluam vocabulário novo, apenas é necessário adicionar os termos nas produções 'tipo', 'acao' e 'keyword'.

```
bcQAS: par+
     ;

par: '(' intencao ';' resposta')'
      ;

intencao: tipo ',' acao ',' keywords
        ;
```

A partir do excerto da GIC acima, é possível constatar que a base de conhecimento é constituída por pares, cada par com 2 campos separados por um ponto e virgula. O primeiro ('intencao') permite identificar a que perguntas a resposta se pode associar. O segundo ('resposta') contém a frase que deve ser apresentada aos utilizadores caso o sistema considere que a questão colocada está relacionada com este conhecimento. A base de conhecimento é identificada inicialmente pela palavra reservada "BC:". Alguns exemplos de pares pergunta-resposta sintaticamente corretos que podem ser adicionados são:

```
BC:
( Quando , pagar , [ propinas ] ; ' Até dia 23 de julho .' )
( Qual , ser , [ email , diretor ,  curso ] ; ' ddd@ddd.com. ' )
```

'questoes': define a estrutura das perguntas a responder. Cada questão é composta por palavras que o sistema reconhece, (tipos, acções e *keywords*) que são usadas no processo de obtenção de respostas, por palavras que o sistema não conhece (que podem ser também tipos, ações e keywords com *cases* diferentes, noutros tempo verbais, ou então palavras irrelevantes como por exemplo pronomes), por fim, terminadas com sinais de pontuação (um ou mais pontos de interrogação, exclamação ou finais).

```
questoes: questao+
        ;

questao: (PALAVRA | tipo | acao | keyword )+ PONTOTERMINAL+
       ;
```

A secção das questões, para ser reconhecida corretamente, deve ser inicializada pela palavra reservada "QUESTOES:", tal como definido na gramática. É possível observar na figura 1 vários exemplos de questões sintaticamente corretas.

Encontra-se em anexo na secção A a Gramática Independente de Contexto para observação mais detalhada.

5 Gramática de Atributos

De forma a pôr o trabalho funcional, isto é, criar os mecanismos de resposta procedeu-se à transformação da gramática de uma GIC (gramática independente de contexto) para uma GA (gramática de atributos).

A criação da base de conhecimento é feita através da produção '*bcQAS*'. Os pares pergunta-resposta são lidos do ficheiro de teste e guardados num *hashmap* cujas chaves são os 'tipos' das intenções (e.g. 'Como','Quando', ...) e os valores são Arraylists de pares (intencao, resposta). Um objeto do tipo `Par` contém o tipo, a ação, as keywords e informação associada, correspondendo essencialmente a um par pergunta-resposta da base de conhecimento. Esta decisão foi tomada para tornar a procura mais eficiente.

```
bcQAS returns [HashMap<String, ArrayList<Par>> bc]
@init{$bcQAS.bc = new HashMap<String,ArrayList<Par>>();}
    : t1=par [$bcQAS.bc] (t2=par [$t1.bcOut] {$t1.bcOut = $t2.bcOut;})*
    ;

par [HashMap<String, ArrayList<Par>> bcIn]
    returns [HashMap<String, ArrayList<Par>> bcOut]
    : '(' intencao ';' resposta')'
    {
        $intencao.p.resposta = $resposta.val;
        ArrayList<Par> aux = $bcIn.get($intencao.p.tipo);
        if(aux==null) aux = new ArrayList<Par>();
        aux.add($intencao.p);
```

```
        $bcIn.put($intencao.p.tipo,aux);
        $bcOut = $bcIn;
    }
    ;

intencao returns [Par p]
        : tipo ',' acao ',' keywords
        {
            $intencao.p = new Par();
            $intencao.p.tipo = $tipo.val;
            $intencao.p.acoes = $acao.list;
            $intencao.p.keywords = $keywords.list;
        }
        ;
```

Por outro lado, é importante também nos debruçarmos sobre o processo de resposta.

A produção responsável por responder às questões colocadas é a '*questao*'. Aqui são reconhecidas as várias palavras que compõem uma pergunta que podem já existir na base de conhecimento, nomeadamente, tipos, ações e keywords, mas também palavras que não sejam conhecidas. Cada questão é composta por pelo menos uma palavra e terminada por pelo menos um ponto terminal.

```
questao [HashMap<String, ArrayList<Par>> bc]
@init{
    ArrayList<String> tipos = new ArrayList<String>();
    ArrayList<String> acoes = new ArrayList<String>();
    ArrayList<String> keywords = new ArrayList<String>();
    ArrayList<String> palavras = new ArrayList<String>();
    StringBuffer question = new StringBuffer();
}
    :( PALAVRA { palavras.add($PALAVRA.text);
                 question.append($PALAVRA.text).append(" ");
               }
    | tipo { tipos.add($tipo.val);
             question.append($tipo.val).append(" ");
           }
    | acao { acoes.add($acao.val);
             question.append($acao.val).append(" ");
           }
    | keyword { keywords.add($keyword.val);
                question.append($keyword.val).append(" ");
              }
     )+ (PONTOTERMINAL {question.append($PONTOTERMINAL.text);} )+

    {getAnswer($bc,question,tipos,acoes,keywords,palavras);}
    ;
```

Quando é reconhecida uma pergunta completa, procede-se à tentativa de obtenção de resposta. Esta fica ao encargo do método `getAnswer` definido no `@members`. Esta

tenta obter um match completo das palavras extraídas da questão, com a parte da intenção dos pares (intencao, resposta) que contêm o conhecimento. Se isto não for possível, tenta otimizar o mais possível.

Exemplificando, para a base de conhecimento abaixo os resultados esperados seriam os seguintes.

```
BC:
( Quando , pagar , [ propinas ] ; ' Até dia 23 de julho .' )
( O que , ser , [ ECTS ] ; ' European Credit Transfer System .' )
( Qual , ser , [ email , diretor ,  curso ] ; ' ddd@ddd.com. ' )
( Como , aceder , [ calendário , escolar ] ;
       ' O calendário escolar está disponível no portal académico. ' )
( Como , imprimir , [ calendário , escolar ] ;
       ' O calendário escolar pode ser imprimido através do portal académico. ' )
( Onde , inscrever , [ exame , época , especial ] ;
       ' Incrições para exame de Época Especial são feitas no portal académico. ' )
( Onde , inscrever , [ exame , época , recurso ] ;
       ' Não é necessária incrição para realizar exame de recurso. ' )


Gostava de saber quando se pagam as propinas de 2018/19 ...
R0:' Até dia 23 de julho .'

Siri, diz-me: O QUE SÃO ECTS ?!
R0:' European Credit Transfer System .'

Qual é o nome do reitor ?
Não foi encontrada resposta à sua pergunta.

ola, onde nos devemos inscrever para o exame de Época Especial ?
R0:' Incrições para exame de Época Especial são feitas no portal académico. '

Onde nos devemos inscrever para o exame de Época de Recurso ?
R0:' Não é necessária incrição para realizar exame de recurso. '

Onde nos devemos inscrever para o exame de recurso ?
R0:' Não é necessária incrição para realizar exame de recurso. '

Onde nos devemos inscrever para o exame ?
R0:' Incrições para exame de Época Especial são feitas no portal académico. '
R1:' Não é necessária incrição para realizar exame de recurso. '

Como aceder para imprimir o calendário escolar ?
R0:' O calendário escolar está disponível no portal académico. '
R1:' O calendário escolar pode ser imprimido através do portal académico. '
```

A Gramática de Atributos e versão final do *Q&A System* com área de conhecimento sendo a FAQ da Universidade do Minho pode ser visualizada em anexo na secção B.

6 Conclusão

Com a realização deste trabalho, foi-nos possível concluir que as várias funcionalidades proporcionadas pelas Gramáticas de Atributos (a atribuição de valores a atributos associados às produções de uma gramática formal) permitem a construção de programas e linguagens bastante poderosas de forma simples, visualmente legível e de fácil compreensão.

Partindo destas vantagens fornecidas, fomos capazes de desenvolver um programa bem consolidado que permite, a partir de conhecimento prévio acerca de uma área específica, responder a várias questões colocadas pelo utilizador envolvendo múltiplas áreas e serviços da Universidade do Minho.

Contudo, uma das melhorias que poderia ser feita no futuro ao programa desenvolvido é o carregamento do vocabulário a ser usado na base de conhecimento e.g tipos (Qual, Como, ...), ações (imprimir, inscrever,...) e as *keywords* (exame, reitor,...) ser feito a partir do ficheiro de teste. Isto permitiria uma maior flexibilidade, sendo possível reaproveitar a mesma gramática para várias áreas de conhecimento diferentes, bastando apenas usar ficheiros de teste diferentes.

Referências

[1] *FAQ*. `https://dictionary.cambridge.org/us/dictionary/english/faq`. Acedido em 03 de Janeiro de 2019.

[2] *Q&A software*. `https://en.wikipedia.org/wiki/Q%26A_software`. Acedido em 03 de Janeiro de 2019.

A Gramática Independente de Contexto

```
grammar QASystemGIC;

qas: 'BC:' bcQAS 'QUESTOES:' questoes
   ;

questoes: questao+
        ;

questao: (PALAVRA | tipo | acao | keyword )+ PONTOTERMINAL+
       ;

bcQAS: par+
     ;

par: '(' intencao ';' resposta')'
      ;

intencao: tipo ',' acao ',' keywords
        ;
```

```
resposta: TEXTO
        ;

tipo: 'Porquê' | 'O que' | 'Quando' | 'Onde' | 'Como'
    ;

acao: 'aceder' | 'imprimir' | 'inscrever' | 'pagar' //....
    ;

keywords: '[' keyword ( ',' keyword)* ']'
        ;

keyword: 'propinas' | 'época' | 'especial' | 'portal'
       | 'académico' | 'Universidade' | 'Minho' //....
       ;

/* Definição do Analisador Léxico */
TEXTO:    (('\'') ~('\'')* ('\'));

fragment LETRA : [a-zA-ZáéíóúÁÉÍÓÚÃãÕõâêôÂÊÔÀÈÌÒÙàèìòùÇç] ;

fragment DIGITO: [0-9];

fragment SIMBOLO : [-%$\euro@&()\[\]:{}=><+*;,ºª~^/\'"];

PONTOTERMINAL: [?.!];

PALAVRA: (LETRA | DIGITO | SIMBOLO)+;

Separador: ( '\r'? '\n' | ' ' | '\t' )+  -> skip;
```

B Gramática de Atributos

```
grammar QASystemGA;

@header{
    import java.util.*;
    import java.util.stream.Collectors;
}

@members{
    class Par{
        String tipo;
        ArrayList<String> acoes;
        ArrayList<String> keywords;
        String resposta;

        public String toString(){
             StringBuffer sb = new StringBuffer();
```

```
          sb.append("t = ");
          sb.append("("+this.tipo+",");
          sb.append(Arrays.toString(this.acoes.toArray())+",");
          sb.append(Arrays.toString(this.keywords.toArray())+")");
          return sb.toString();
    }
}

boolean containsElemList(ArrayList<String> l1, ArrayList<String> l2, int n){
    boolean ret=false;
    int size1 = l1.size();
    int size2 = l2.size();
    int matches=0;

    for(int i=0; i<size1 && !ret; i++)
        for(int j=0; j<size2 && !ret; j++) {
            ret=l2.get(j).toLowerCase().equals(l1.get(i).toLowerCase());
            matches++;
        }
    if(matches>n) n = matches;
    if (matches<n) return false;
    return ret;
}

boolean containsAllKeywords(ArrayList<String> l1, ArrayList<String> l2){
    boolean ret=false;
    int size1 = l1.size();
    int size2 = l2.size();
    int nkeys = l2.size();

    if(size2==0) return false;

    for(int i=0; i<size1 && nkeys>0; i++)
        for(int j=0; j<size2 && nkeys>0; j++)
            if (l2.get(j).toLowerCase().equals(l1.get(i).toLowerCase()) == true) {
                nkeys--;
            }
    if(nkeys==0) return true;
    else return false;
}

/* cria lista com as keywords presentes na BC*/
ArrayList<String> addkeywords (ArrayList<String> l, ArrayList<Par> t) {
    ArrayList<String> k = new ArrayList<String>();

    for (Par pair : t)
        for (String s : pair.keywords)
            if(!l.contains(s)) l.add(s);

    return l;
}
```

```
/* Obtém a(s) resposta(s) para cada questão */
void getAnswer(HashMap<String, ArrayList<Par>> bc, StringBuffer question,
               ArrayList<String> tipos, ArrayList<String> acoes,
               ArrayList<String> keywords, ArrayList<String> palavras){

    int tipoSize = tipos.size();
    ArrayList<String> keywordsBC = new ArrayList<String>();
    ArrayList<String> keywordsPalavras = new ArrayList<String>();
    ArrayList<String> resp;
    ArrayList<String> respteste;
    ArrayList<Par> aux = new ArrayList<Par>();


    for(ArrayList<Par> l : bc.values())
        keywordsBC = addkeywords(keywordsBC,l);

    if(tipoSize>0){
        for(int i=0;i<tipoSize;i++)
            aux.addAll(bc.get(tipos.get(i)));
    }else{
        for(ArrayList<Par> l : bc.values())
            aux.addAll(l);
    }

    for (String s : palavras) {
        if(keywordsBC.contains(s.toLowerCase()))
            keywordsPalavras.add(s);
    }

    if(keywordsPalavras.isEmpty())
        for(String s : keywords)
            keywordsPalavras.add(s);

    int x=0;
    respteste = aux.stream()
       .filter(a -> containsElemList(a.acoes,acoes,x) || containsElemList(a.acoes,palavras,x))
       .filter(a -> containsAllKeywords(a.keywords,keywords) && containsAllKeywords(a.keywords,keywordsPalavras))
       .filter(a -> containsAllKeywords(a.keywords,keywordsPalavras))
       .map(a -> a.resposta)
       .distinct()
       .collect(Collectors.toCollection(ArrayList::new));

    if(!respteste.isEmpty()) {

        System.out.println("\n"+question.toString());
        int w=0;
        for(String r : respteste)
            System.out.println("R" + w++ + ":" + r);

    } else {

        int n=0;
```

```
            resp = aux.stream()
                      .filter(a -> containsElemList(a.acoes,acoes,n) || containsElemList(a.acoes,palavras,n))
                      .filter(a -> containsElemList(a.keywords,keywords,n) || containsElemList(a.keywords,palavras,n))
                      .map(a -> a.resposta)
                      .distinct()
                      .collect(Collectors.toCollection(ArrayList::new));

            System.out.println("\n"+question.toString());
            int w=0;
            for(String r : resp)
                System.out.println("R" + w++ + ":" + r);

            if (resp.isEmpty()) System.out.println("Não foi encontrada resposta à sua pergunta.");
        }
    }
}

qas: 'BC:' bcQAS 'QUESTOES:' questoes [$bcQAS.bc]
   ;

questoes [HashMap<String, ArrayList<Par>> bc]: (questao [$questoes.bc])+
                                                ;

questao [HashMap<String, ArrayList<Par>> bc]
@init{
    ArrayList<String> tipos = new ArrayList<String>();
    ArrayList<String> acoes = new ArrayList<String>();
    ArrayList<String> keywords = new ArrayList<String>();
    ArrayList<String> palavras = new ArrayList<String>();
    StringBuffer question = new StringBuffer();
}
    : (PALAVRA {palavras.add($PALAVRA.text); question.append($PALAVRA.text).append(" ");}
    | tipo {tipos.add($tipo.val); question.append($tipo.val).append(" ");}
    | acao {acoes.add($acao.val); question.append($acao.val).append(" ");}
    | keyword {keywords.add($keyword.val); question.append($keyword.val).append(" ");} )+
    (PONTOTERMINAL {question.append($PONTOTERMINAL.text);} )+
    {getAnswer($bc,question,tipos,acoes,keywords,palavras);}
    ;

bcQAS returns [HashMap<String, ArrayList<Par>> bc]
@init{$bcQAS.bc = new HashMap<String,ArrayList<Par>>();}
    : t1=par [$bcQAS.bc] (t2=par [$t1.bcOut] {$t1.bcOut = $t2.bcOut;})*
    ;

par [HashMap<String, ArrayList<Par>> bcIn] returns [HashMap<String, ArrayList<Par>> bcOut]
    : '(' intencao ';' resposta')'
    {
        $intencao.p.resposta = $resposta.val;
        ArrayList<Par> aux = $bcIn.get($intencao.p.tipo);
        if(aux==null) aux = new ArrayList<Par>();
        aux.add($intencao.p);
        $bcIn.put($intencao.p.tipo,aux);
```

```
        $bcOut = $bcIn;
    }
    ;

intencao returns [Par p]
        : tipo ',' acao ',' keywords
        {
            $intencao.p = new Par();
            $intencao.p.tipo = $tipo.val;
            $intencao.p.acoes = $acao.list;
            $intencao.p.keywords = $keywords.list;
        }
        ;

resposta returns [String val]
        : TEXTO {$resposta.val = $TEXTO.text;}
        ;

tipo returns [String val]
    : ( t='Porquê' | t='O que' | t='Quando' | t='Onde' | t='Como' | t='Qual' | t='Quem') {$tipo.val = $t.text;}
    ;

acao returns [ArrayList<String> list, String val]
@init{$acao.list = new ArrayList<String>();}
     : 'aceder' {$acao.list.add("aceder"); $acao.list.add("acedo"); $acao.val="aceder";}
     | 'imprimir' {$acao.list.add("imprimir"); $acao.list.add("imprimo"); $acao.val="imprimir";}
     | 'ser' {$acao.list.add("é"); $acao.list.add("foi"); $acao.list.add("são");}
     | 'inscrever' {$acao.list.add("inscrever"); $acao.list.add("inscrevo"); $acao.val="inscrever";}
     | 'pagar' {$acao.list.add("pagar"); $acao.list.add("pago"); $acao.list.add("pagam"); $acao.val="pagar";}
     | 'tem' {$acao.list.add("tem"); $acao.list.add("teve"); $acao.list.add("tinha");}
     | 'haver' {$acao.list.add("há"); $acao.list.add("havia"); $acao.list.add("houve");}
     | 'funcionar' {$acao.list.add("funcionar"); $acao.list.add("funciona"); $acao.list.add("funcionará");}
     ;

keywords returns [ArrayList<String> list]
@init{$keywords.list = new ArrayList<String>();}
        : '[' k1=keyword {$keywords.list.add($k1.val);} ( ',' k2=keyword {$keywords.list.add($k2.val);})* ']'
        ;

keyword returns [String val]
       : ( t='propinas' | t='época' | t='especial' | t='email' | t='diretor' | t='curso'
       | t='portal' | t='académico' | t='Universidade' | t='Minho' | t='exame' | t='recurso'
       | t='calendário' | t='escolar' | t='reitor' | t='horário' | t='regulamento'
       | t='código' | t='ético' | t='conduta' | t='direitos' | t='deveres' |  t='cadeira'
       | t='unidade' | t='curricular' | t='ECT' | t='ECTS' | t='Erasmus') {$keyword.val=$t.text;}
       ;

/* Definição do Analisador Léxico */
TEXTO: (('\'') ~('\'')* ('\''));

fragment LETRA : [a-zA-ZáéíóúÁÉÍÓÚÃãÕõâêôÂÊÔÀÈÌÒÙàèìòùÇç] ;
```

```
fragment DIGITO: [0-9];

fragment SIMBOLO : [-%$€@&()\[\]:{}=><+*;,ºª~^/\'"];

PONTOTERMINAL: [?.!];

PALAVRA: (LETRA | DIGITO | SIMBOLO)+;

Separador: ( '\r'? '\n' | ' ' | '\t' )+  -> skip;
```