Capítulo 1

Utilização da programação declarativa para processamento do CETEMPúblico

Agostinho Monteiro, Júlio Barbas e Nuno C. Marques

Luís Costa, Diana Santos e Nuno Cardoso, editores, *Perspectivas sobre a Linguateca / Actas do encontro Linguateca: 10 anos*, Capítulo 1, p. 1–5, 2008

Várias entidades têm desenvolvido um substancial esforço na elaboração e processamento de corpora de texto, com utilização de diferentes linguagens, ambientes de desenvolvimento e formatos de representação. Para tal, cada entidade envolvida no processo utilizou os recursos que entendeu serem mais adequados ao seu trabalho com a consequente redundância de repositórios e eventual ambiguidade nas classificações.

Bom seria que existisse um repositório principal, de acesso livre, criteriosamente actualizado por diferentes ferramentas e validado por entidades para tal autorizadas. O trabalho desenvolvido pela Linguateca é um bom exemplo neste sentido. Nesta comunicação propõe-se um novo formato que possibilita a utilização directa da programação declarativa. Esta proposta deve ser integrada nas soluções e formatos existentes, com o objectivo de potenciar a interoperabilidade, que pode existir entre diferentes aplicações, já existentes ou a desenvolver entendidas como serviços, que utilizem e manipulem os textos. Para tal, pode-se seguir uma filosofia SOA (*Service Oriented Architecture*), cujo principal objectivo é precisamente a reutilização de aplicações já existentes e a sua integração com outras a desenvolver. Neste quadro, as aplicações já existentes devem ser entendidas como serviços, mediante disponibilização de um canal comum que liga, transparentemente, o requerente de um serviço ao seu fornecedor. A arquitectura SOA define igualmente conjuntos de métricas que devem ser aplicados aos diversos serviços, para que seja possível a análise da sua utilidade e desempenho. A Linguateca pode constituir o referido canal SOA, disponibilizando, como já o faz, o CETEMPúblico (Rocha e Santos, 2000) no formato nativo e em XML, sendo este último uma escolha consensual, por ser suficientemente expressivo, como formato de partilha de informação. No entanto, a escolha do XML deixa em aberto que tipo de ferramentas utilizar para o tratamento da informação e qual a melhor estrutura a utilizar para representar a informação a ser tratada com essas ferramentas. Para tentar responder a estes problemas, apresenta-se um formato lógico, implementado com base no PROLOG (mais especificamente no SWI-PROLOG (Wielemaker, 2008)) que resolve ambas estas questões de forma elegante e integrada. Foram utilizados como casos de estudo dois corpora de domínio público distintos: o corpus *Susanne* (Sampson, 1995) para o inglês e a *Floresta Sintá(c)tica* (Bick et al., 2007) (`www.linguateca.pt/Floresta`), mais especificamente o subconjunto anotado da Floresta também incluido no CETEMPúblico, o Bosque.

A motivação para a utilização do PROLOG e linguagens lógicas suas derivadas (por exemplo o DyALog (`dyalog.gforge.inria.fr`) advém do facto de, sendo o PROLOG uma linguagem declarativa, ser extremamente fácil descrever o conhecimento e efectuar inferências e deduções sobre esse conhecimento. Assim, a programação declarativa, na linguística computacional, desde cedo tem sido considerada uma ferramenta clássica. Por exemplo, as Gramáticas de Cláusulas Definidas[1] do PROLOG (Pereira e Shieber, 1987) são ferramentas extremamente úteis para a expressão de gramáticas e para a sua aplicação. Es-

[1] *DCG — Definite Clause Grammars*

tas gramáticas podem ser utilizadas para extrair uma representação lógica relativamente a determinados padrões de etiquetas que ocorrem em textos previamente marcados.

Contudo, apesar de o PROLOG ser excelente para a manipulação simbólica, as suas capacidades para tratamento *Input/Output* e para análise sub-simbólica eram inicialmente mais limitadas. Talvez por isso, em geral, o PROLOG tenha sido menos utilizado para representação de corpora de textos. As implementações mais recentes do PROLOG resolvem este problema de forma fácil e elegante ao possibilitar a integração com o XML.

1.1 Descrição do formato TXT/2

Na estrutura proposta (designada formato $TXT/2$), parte-se do princípio que, num texto, as frases são naturalmente divididas em palavras. No entanto, aqui, generaliza-se o conceito de palavra para estruturas de palavras (w), cada uma recursivamente composta por sequências de palavras (cw - *container of* w). Com base neste conceito de palavra/estrutura de palavras ganhamos uma maior abstracção, sem nunca perder a informação original do texto em causa. Assim, uma estrutura $TXT/2$ é constituída por estruturas de palavras (w), aglomeradas numa lista PROLOG. Esta lista representa todo o conhecimento necessário para o processamento do texto. Cada uma das palavras é caracterizada por vários atributos, salientando-se os atributos que indicam a palavra (wd) e a etiqueta dessa palavra (tag – para etiquetas morfossintácticas). Da mesma forma que se associa uma etiqueta a cada palavra, também se pode associar uma etiqueta a uma sequência de palavras cw. Nesse caso podemos igualmente ter a tag da palavra para terminais ou um atributo $gtag$ – para não terminais, tipicamente contendo a etiqueta sintáctica. Note-se que deve ser atribuído um *ID* único a cada elemento do texto. Um exemplo característico é o caso da palavra composta "Presidente da República", cuja notação (por questões de espaço, forçosamente simplificada) seria a seguinte:

```
txt(1, [w([1,wd='Presidente da República', semantic=headOfState, tag='PROP',
          cw=[w([1-1, wd='Pesidente', bw=presidente,gen=masc]),
                  w([1-2, wd='da',
                          cw=[w([1-2-1, wd='de', tag='PREP']),
                          w([1-2-2, wd='a', tag='DET', gen=fem])]]),
                    w([1-3, wd='República', gen=fem])]])
          ]).
```

No exemplo apresentado, todo o contexto (que, consoante o caso, poderá ser um texto ou simplesmente uma frase) deverá ser representado na lista PROLOG. O primeiro argumento deste termo é um *ID* único desse texto (facilitando a interligação com uma base de dados relacional). Como foi referido, a lista $TXT/2$ é composta por palavras (cada uma

com um *ID* especifico, discriminante relativamente ao texto em que se insere), inserindo os elementos cw, recursivamente, outras listas deste tipo.

Uma das grandes vantagens desta abordagem é a sua fácil e directa integração com as *D*CGs do PROLOG. Desta forma é possível representar e gerar árvores de análise e gramáticas para extracção de conhecimento dos textos anotados segundo a formatação aqui proposta. Eis o exemplo gerado para a primeira árvore no Bosque:

```
txt(s1,[
        w([s1_500, gtag=s,
         ref="CP1-1", source="CETEMPúblico n=1 sec=clt sem=92b", forest="1",
         text="Um revivalismo refrescante",
           cw=[w([s1_501, gtag=np,
                    cw=[w([s1_1, wd='Um', tag=art, label='>N', lemma=um,
                    morph='M S', extra=arti, extra2= --, extra3= --]),
                w([s1_2, wd=revivalismo, tag=n, label='H',
                    lemma=revivalismo, morph='M S', extra= --, extra2= --,
                        extra3= --]),
                        w([s1_502, gtag=adjp,
                        cw=[w([s1_3, wd=refrescante, tag=adj, label='H',
                             lemma=refrescante, morph='M S', extra= --,
                             extra2= --, extra3= --])],label='N<'])],
              label='UTT'])]])]).
```

Note-se em particular os elementos tag, terminais da gramática, classificadores, e os elementos $gtag$ que, pelas suas características não terminais, são susceptíveis de desenvolvimento arborescente, como cw, até se encontrarem elementos terminais. As anotações extra, como $label$, $lemma$ ou $morph$ são mantidas na representação proposta não se perdendo assim nenhuma informação.

Outra vantagem é a gestão de co-referências. Assim, neste formato, podem facilmente ser incluídas co-referências bem como qualquer outro tipo de anotações, atributos ou variáveis. A título de exemplo considere-se a frase "O João, que precisava de um livro, foi à biblioteca" na qual a componente relativa "que precisava de um livro" pode ser representada recursivamente.

1.2 Exemplos de aplicação e resultados

A recursividade na estrutura $TXT/2$ pode ser facilmente tratada com o PROLOG. Assim, por exemplo, para sabermos se uma palavra W está ou não marcada com uma tag (o que torna imediata a construção de um dicionário utilizando o predicado $findall/3$), basta o seguinte código PROLOG:

```
get_wd_tag(W,Tag, IDW, TXT2)  :-
          member(w( [IDW|WL]), TXT2), member(wd=W, WL), member(tag=Tag, WL).
get_wd_tag(W, Tag, IDW, TXT2) :-
          member(w([IDW|WL]), TXT2), member(cw=CW, WL),
          get_w_tag(W, Tag, CW).
```

A primeira cláusula apenas refere que uma palavra W marcada com etiqueta tag é um membro de uma palavra (ela própria membro de uma estrutura $TXT/2$). A segunda cláusula resolve a recursividade, indicando que a estrutura $TXT/2$ pode ser ela própria membro de um elemento cw.

Já o processo inverso à construção do dicionário, i.e. anotação do texto (em inglês, *POS tagging*), requer a utilização do contexto para desambiguar entre as possíveis etiquetas que uma palavra pode ter. Como foi referido, o contexto está disponível na lista $TXT/2$. Assim, tal como em qualquer outra análise sintáctica ou semântica, a facilidade de acesso directo e sequencial às palavras é essencial. Neste momento, um sistema neuro-simbólico, utilizando uma rede neuronal treinada com um conjunto de apenas 5000 palavras extraídas do Bosque Sintáctico e com auxilio de regras de desambiguação (Marques et al., 2007), já consegue obter uma precisão de classificação na ordem dos 94% no CETEMPúblico. No entanto espera-se que este valor melhore substancialmente com a adição de mais características e regras de desambiguação, visto que no *Susanne* este valor já chega aos 95%.

Uma vez que o formato proposto apresenta uma estrutura lógica (compatível com a linguagem PROLOG), é possível a sua conversão para o formato TIGER-XML (TIGER), e vice-versa. Assim, no sentido de apresentar este formato e ferramentas PROLOG associadas como possíveis serviços numa possível arquitectura SOA para a Linguateca, foi igualmente implementado um leitor de informação TIGER-XML, que possibilita a conversão de textos para o formato $TXT/2$. Utilizaram-se como base de estudo os textos em português do Bosque em formato TIGER-XML e o Susanne, convertido para formato TIGER-XML. Mas poderão igualmente ser utilizadas outras formas de converter informação para PROLOG. A título de exemplo, durante o trabalho efectuado foi comum a utilização de informação em formato *S*QL e a utilização da linguagem de reconhecimento de padrões *awk* (Aho et al., 1988).

Note-se ainda que a estrutura $TXT/2$ funciona como um armazém de anotações efectuadas sobre o texto. Isto estimula a interoperabilidade que pode haver entre diversas aplicações que utilizem e manipulem os textos, promovendo uma filosofia *S*OA. Esta filosofia pode funcionar tanto ao nível de módulos internos PROLOG, como ao nível mais geral de *W*ebServices utilizando XML. Assim, a utilização de uma estrutura deste tipo pode contribuir para a utilização de módulos com maior poder dedutivo potenciando, em simultâneo, a partilha de informação que se deseja obter com o projecto da Linguateca.