Especificação e tratamento de Dicionários

José João Dias de Almeida
Departamento de Informática
Universidade do Minho
Campus de Gualtar, 4700 Braga
jj@di.uminho.pt

Resumo

A definição de dicionários é uma tarefa complexa e trabalhosa que permite construir uma base de informação que pode ser usada para vários fins e em várias ambientes diferentes da tradicional impressão em papel.

Essa reutilização é dificultada por um excesso de detalhes tipográficos e pela falta de definições formais e de estrutura que normalmente acompanham o processo de construção.

Neste artigo descreve-se um estudo experimental de especificação e tratamento de dicionários envolvendo:

- Especificação de uma estrutura lógica a usar no dicionário: Descrição formal de cada entrada
- Tratamento de engenharia reversa de um "texto" de dicionário no sentido de construção parcial da referida estrutura
- definição e implementação de um ambiente de consulta e construção de dicionários com:
  - consulta, edição, do dicionário construído
  - uso de analisador morfológico para tratamento de palavras desconhecidas ou derivadas
  - inferência (elementar) de partes da estrutura
  - possibilidade de extracção automática de frases exemplo a partir de córpora
  - possibilidade de programação de "travessias" do dicionário
- definição de uma linguagem (de programação) de definição de dicionários

Índice

1 Introdução

A construção de dicionários é uma tarefa complexa que exige uma enorme disciplina formal dificilmente conseguida sem um projecto elaborado rigorosamente e suportado por ferramentas que ajudem à validação.

O panorama actual, evidencia que a informação contida em dicionários tem que ser estruturada dum modo versátil e reutilizável levando à preocupação de separar a definição do uso que lhe vai ser dado (Sperberg-McQueen and Burnard, 1994, SGML/TEI).

A evolução da linguística e a utilização de técnicas das ciências da computação, da inteligência artificial e das ciências documentais vieram salientar a importância de questões como classificação, tipo, normalização e herança. A organização da informação orientada ao conceito (Miller et al., 1993, WordNet) e as aplicações associadas a PLN (Krieger and Schäfer, 1994; Krieger, 1995, TFS) (Calzolari, 1993), demonstram a necessidade de poder ver os dicionários como bases de conhecimentos a serem usadas como módulos de software capazes de oferecer uma funcionalidade variada em cooperação com outras peças formando aplicações.

Neste documento refere-se um trabalho experimental de construção e processamento de dicionários envolvendo:

- especificação formal da estrutura do dicionário (ver secção 2)
- tratamento em engenharia reversa de um dicionário de modo a construir uma base de conhecimento (ver secção 3)
- definição de uma aplicação "dicionário" que agrupa a base de conhecimento "dic" com um modulo de análise morfológica "jspell" e com um "pesquisador" de padrões em córpora (NLgrep) (ver secção 4)

No sentido de facilitar a criação de dicionários, apresenta-se ainda exemplos de uma linguagem de programação de dicionários (ver secção 5).

2 Estrutura lógica

A definição de uma estrutura lógica de um dicionário abstrai dos detalhes pictóricos/operacionais correspondentes às utilizações da informação nele contida.

Essa definição deve ser feita de um modo formal de modo a fugir às ambiguidades que levam a que as interpretações de cada pessoa sejam diversas. A especificação formal da estrutura dum dicionário (ou de partes dela) constitui uma boa base de discussão bem como uma boa base para a construção de ferramentas de validação de partes da informação.

Neste texto apresentamos apenas uma visão superficial desta tarefa, tentando apresentar alguns exemplos no sentido de ilustrar alguns dos problemas/conclusões inerentes à especificação.

2.1 Estrutura lógica usando CAMILA

CAMILA é uma linguagem de especificação formal criada na Universidade do Minho, baseada em conjuntos e usada para descrição e prototipagem de um grande leque de objectos incluindo, por exemplo, programas, redes de emergência, representação de conhecimento histórico, etc.

Para uma completa descrição de CAMILA ver (Barbosa and Almeida, 1995). Em CAMILA a definição de entidades é feita indicando a sua estrutura (conjunto portador) e se necessário, uma propriedade (o invariante) que descreve restrições semânticas a serem verificadas pelos elementos válidos.

Considere-se um dicionário com a seguinte definição:

```
Dic = Palavra -> Inf | Inf_derivada
```

ou seja, um dicionário (`Dic`) estabelece uma correspondência entre palavras e informação associada; `Inf` corresponde a uma entrada normal e `Inf_derivada` corresponde à definição de uma palavra derivada.

```
Inf = Fon × Cla × Uso × Gra × Sem × Der × ...
Inf_derivada = Rad:Palavra × Cla × Exe:frase × Tra:frase
```

A seguinte entrada exemplifica o uso de `Inf_derivada` :

```
acusação  Rad= acusar
          Cla= n.f.
          Exe= ... fez a acusação ...
          Tra= ... acusou ...
```

Um invariante natural seria garantir que os radicais das palavras derivadas constem do dicionário

$$invDic(d) = \forall_{x \in dom(d)} \cdot derivada(d_x) \Rightarrow rad(d_x) \in dom(d)$$

Há noções menos explícitas como a de acepção que surgem claramente quando tentamos especificar partes de certos dicionários.

Por exemplo o campo `Uso`, no caso do dicionário do Prof. Mário Vilela (Vilela, 1991), na sua situação mais complexa, pode ser descrito por:

```
Uso      = Acepção -> Fra_a
Fra_a    = (Cla × assinatura) -> Exemplos
Exemplos = Nº -> frase
```

tanto `Acepção` como `Nº` são necessários para que possam ser usados noutros campos para referenciar exemplos e acepções. `Assinatura` corresponde a um padrão abstracto de concordância (que não aprofundaremos neste texto).

No exemplo seguinte as acepções são descritas por números romanos e os exemplos usam números inteiros.

```
{ I => { (v.t. × (pessoa) acusa pessoa de coisa) =>
             { 1 => ... acusou ... de roubo
               2 => ... ... ... acusa ...
             }
         (v.t. × (pessoa) acusa pessoa de adj.) =>
             { 1 => ... acusamos ... de corrupto
               ...
             }
       }
 II => { (v.t. × (pessoa) acusa coisa) =>
             { 1 => já acusaste a recepção da carta
             }
       }
}
```

Tal como acontece em (Vilela, 1991), esta estrutura (e respectiva notação) fica simplificada para situações em que só haja uma acepção, uma classificação ou uma *assinatura*.

A especificação de "uso" evidencia a noção de acepção, de acepção contextualizada e ainda de exemplo dentro de uma acepção contextualizada. Paralelamente, a definição de sinónimos, antónimos, etc. está associada a estas noções (Ex: *acusa* na acepção II tem como sinónimo *dar conhecimento*).

3 Tratamento do dicionário em engenharia reversa

A tarefa de tratamento reverso do dicionário consistiu em construir um programa para traduzir um ficheiro contendo comandos tipográficos[1] para uma estrutura Perl correspondente à estrutura lógica (preservando, dentro do possível, a totalidade da informação existente).

Essa tradução baseou-se em programação usando intensivamente expressões regulares, e correspondeu à extracção dos vários campos, subcampos, tomando como base as marcas tipográficas como tipo de letra, tamanho, caracteres especiais, comandos diversos de espaçamento e outros.

Esta tradução foi feita na linguagem Perl (Vromans, 1995) e gerou uma estrutura Perl `DB_File` que estabelece a correspondência entre cada entrada e a informação a ela associada segundo a estrutura parcialmente descrita na secção 2.

Durante o processo de tradução foi tornada explícita informação que aparece dispersa no dicionário impresso (Ex.: listas de contextos).

Pelo facto de se ter passado duma sequência de comandos tipográficos para um objecto dum tipo estruturalmente mais rico foi possível escrever um módulo `dict` para manuseamento desta informação e que permite:

- aceder directamente a cada um dos campos/subcampos
- alterar, editar, acrescentar, visualizar, a totalidade ou partes da informação

[1]responsáveis por mudanças de tipo de letra, tamanho de letra, mudanças de fonte, mudanças de página, de coluna, figuras, etc., proveniente de um editor de texto Macintosh

4 O sistema ETdic

O sistema ETdic (especificação e tratamento de dicionários) agrupa para além do módulo `dict` já referido:

- um analisador morfológico usando por base o JSPELL (Almeida and Pinto, 1994a) para tratamento de palavras derivadas/flexionadas não contidas no dicionário bem como para tratamento de palavras aproximadas. Este módulo tem ainda mecanismos programáveis elementares de inferência que permitem cálculo de certos campos indefinidos.

- um módulo para pesquisa de palavras (e suas derivadas/flexionadas) em conjuntos de frases (NLgrep)

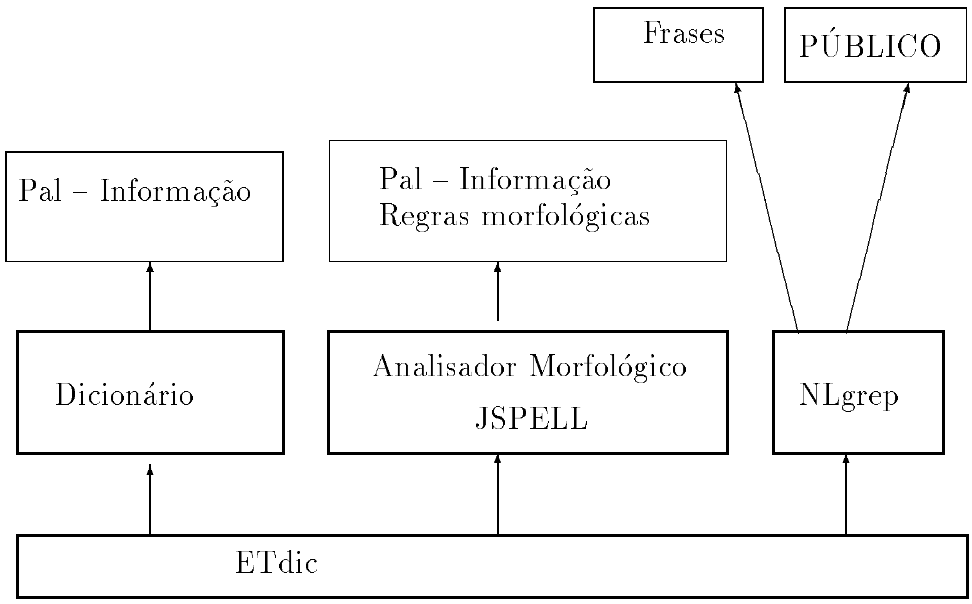

Figura 1: arquitectura do sistema ETdic

Com base no sistema de módulos descrito construiu-se um interpretador ETdic que liga as várias funcionalidades.

Vamos de seguida apresentar algumas características através de exemplos. A ênfase será posta naquilo que mais se afaste do usual.

Para tornar mais claro a origem da informação, o interpretador apresentado vai ter funções muito simples[2] como:

[2]Na versão actual o ETdic, durante a consulta, activa automaticamente a análise morfológica quando a palavra não existe no dicionário, escolhe automaticamente a única opção quando só existe uma, etc.

- procurar palavra no dicionario
- "morf" - fazer analise morfológica da ultima palavra
- "ex" - procurar exemplos da palavra anterior no corpus

4.1 Derivadas de palavras existentes no dicionário

Quando uma palavra não existe no dicionário o analisador morfológico é activado construindo uma lista de possíveis origens da palavra sendo também apresentadas as informações obtidas pelo analisador morfológico.

Todas as informações serão pares atributo-valor.

```
-> comprador

Palavra desconhecida
{  name => comprador}

-> morf

{ _1 => {
      CAT => a_nc
      FSEM => dor
      T => inf
      N => s
      TR => t
      rad => comprar
      G => m}
   name => comprador}
```

se seguidamente escolhermos a hipótese "1" (alias única apresentada) a informação que aparece será

```
-> 1
{  name => comprador
   Sem => [ Aquele ou aquilo que realiza a acção de comprar.]
   TR => t
   rad => {
      Cla => v. tr.
      name => comprar
      Fra => {
         I => [ (pessoa) @ coisa
            1) O Miguel comprou um carro novo.
            2) A Filipa comprou mais um livro.]
```

```
      II => [ (fig.) (pessoa) @ pessoa
         1) O advogado comprou as testemunhas.
         ...]}
   conj => 1
   Der => {
      Comprador => {
         Cla => n. m.
         trad => O José é o comprador daquela mota.
         ex => O José comprou aquela mota. }}
   Fon => [kõprár]
   Sig => [ (pessoa) @ coisa
            (fig.) (pessoa) @ pessoa]
   Sem => { 1 => ...
            2 => ...
G => m
CAT => a_nc
N => s}
```

4.1.1 Preenchimento de campos por análise morfológica

```
-> rebentamento

Palavra desconhecida
{ name => rebentamento}

-> morf

{  _1 => {
      CAT => nc
      FSEM => mento    (1)
      T => inf
      N => s
      TR => t,i        (2)
      rad => rebentar
      G => m}
   name => rebentamento}

-> 1

Palavra desconhecida
{  name => rebentamento
   Sem => [ Acção de rebentar.]
   TR => t,i
```

```
   rad => { name => rebentar}
   G => m
   CAT => nc
   Gra => pode reger a preposição 'de' seguido do elemento que se vai rebentar.
   N => s}
```

O campo `Sem` foi calculado com base em `(1)`.
O campo `Gra` foi calculado com base nos valores de `(1)` e da característica "transitividade" `(2)`.

Durante este processo está a ser feito inferência de valores de campos desconhecidos com base em valores de outro campos[3]. Na versão actual, as regras de inferência são programáveis através de claúsulas da forma

```
exp_booleana => campos a adicionar - campos a retirar
```

A palavra "rebentar" não existe ainda no dicionário.

4.1.2 Enriquecimento com exemplos determinados dinamicamente

Na continuação do exemplo anterior, vão ser pedidos exemplo de utilização de "rebentar" em frases dos exemplos usados no dicionário. Esses exemplos são construídos com base num corpus (ver fig. 1) produzindo o seguinte resultado:

```
-> ex

{  name => rebentamento
   Sem => [ Acção de rebentar.]
   TR => t,i
   rad => {
      Fra => [
         Na Primavera rebentam os botões das flores.
         A corda rebentou com o excesso de peso.
         Quando o corredor entrou no estádio, rebentou uma...
         A câmara fez cortar as ... a rebentar o pavimento.
         Rebentou uma conduta da rede de distribuição de água à cidade.
         O prédio ficou todo inundado, quando a canalização rebentou.]
      name => rebentar}
   G => m
   CAT => nc
   Gra => pode reger a preposição 'de' seguido do elemento que se vai rebentar.
  N => s}
```

[3] A determinação de uma semântica só é possível para certos afixos; para sufixos como "eiro", dado o seu carácter multi-função, a determinação da respectiva semântica é condicionada ao conhecimento de informação de classe.

4.1.3 Exemplo

No exemplo seguinte mostra-se partes do resultado duma consulta da palavra *meta*. O exemplo serve para descrever o uso do analisador morfológico para procurar origens da palavra em causa (*meta* flexionado de *meter*!).

```
-> meta

{  Cla => n. f.
   name => meta
   Fra => {
      I => [
         1) O atleta desistiu a poucos metros da meta.
         ...
   Sem => {
      1 => [
         @ é um \sc{sinal} (poste, marca, cordel, linha...) que
         ...
      2 => [
         \s(II), @ é o \sc{alvo}, a \sc{mira},
         ...
   Fig => \fig(,192)---Qual é a linha da meta  ?

-> morf

{  name => meta
   _1 => { CAT => nc
           N => s
           G => f}
   _2 => { P => 1_3
           CAT => v
           T => pc
           N => s
           rad => meter}
   _3 => { P => 3
           CAT => v
           T => i
           N => s
           rad => meter}
   Cla => n. f.
   Sem => {
      1 => [
         @ é um \sc{sinal} (poste, marca, cordel, linha...) que
       ...
```

```
-> 2
{  name => meta
   T => pc
   N => s
   rad => {
      Cla => v. tr. e prep., v. pron. e prep.
      name => meter
      Sig => [
         [v. tr. e prep.]
         (pessoa) @ coisa {em} lugar
         (pessoa) @ pessoa {em} lugar
         [v. tr.]
         (ser animado) @ dó, medo, pena a pessoa
         [v. pron. e prep.]
         (ser animado) \bf{meter-se em, sob, por}+ n. de lugar
         (pessoa) @-se+a + inf.
         (pessoa) @-se+com ser animado
         (pessoa) @-se+em apuros]
      Sem => {
         1 => [
            meter coisa {em} lugar tem por sins.:
            ...
```

5 Programação de dicionários

Como se disse anteriormente ao juntarmos os módulos dicionário, analisador morfológico, e pesquisador de padrões, constitui-se um núcleo de base para tratamento de dicionários.

Seguidamente apresenta-se exemplos de ”programação” usando esta plataforma de base no sentido de permitir:

- criar abreviaturas/arquétipos que facilitem definições de palavras
- criar funções que insiram conjuntos de palavras associadas

```
fruto(tabaiba,{origem => "zonas quentes, ilha da Madeira"}).

fruto(tomate,{utilidade => "atirar aos maus oradores"}).

{ name => tomate
  Cla => n.m.
```

```
  Sem => { 1 => "tomate é o fruto do tomateiro.
                 O tomate é usado para atirar aos maus oradores" }
  Der => { name => tomateiro
           Cla => n.m.
           Sem => "planta que dá tomates }
}
```

6 Conclusões e trabalho futuro

Quanto ao trabalho de engenharia reversa, os resultados obtidos foram positivos[4]. Sem qualquer intervenção manual foram reconhecidas a totalidade das entradas do dicionário, tendo sido encontradas apenas dois erros (entradas aglutinadas).

A informação correspondente aos gentílicos e ao dicionário do dicionário foram também reconhecidas mas ainda não integradas no sistema.

O trabalho envolvendo ligação do dicionário com o analisador morfológico e com o pesquisador de formas em corpus, vai ser completado com uma camada de interface e com alguns detalhes que tem a ver com um funcionamento mais ergonómico.

Conjuntamente com o trabalho aqui descrito, tem decorrido uma análise específica do campo semântico tendo em vista construir (semi-automaticamente) uma rede semântica análoga à WordNet (Miller et al., 1993).

Agradecimentos

A autor agradece a José Carlos Medeiros pela disponibilização de algum material necessário a este trabalho.

Bibliografia

Almeida, J. Joao and Ulisses Pinto. 1994a. Jspell – um módulo para análise léxica genérica de linguagem natural. In *Actas do Congresso da Associação Portuguesa de Linguística, Évora.*

Almeida, J.João and Ulisses Pinto. 1994b. Manual de utilizador do JSpell. Manual, Universidade do Minho, Julho.

[4]O dicionário (Vilela, 1991) apesar de conter gralhas e incoerências, revela grande perfeição de revisão e consistência sem os quais o trabalho de reconhecimento seria muito dificultado.

Barbosa, Luis and J.Joao Almeida. 1995. *System Prototyping in CAMILA.* University of Minho. Lecture notes for the system Design Course, Computer System Engineering, University of Bristol.

Calzolari, Nicoletta. 1993. Computational lexicons. reader of 5.LLI, Faculdade de Letras, Universidade de Lisboa.

Correia, Margarita. 1994. Bases digitais lexicias na união europeia. In *Simpósio de Lexicologia, Lexicografia e Terminologia.* Araraquara.

Krieger, Hans-Ulrich. 1995. Typed feature formalisms as a common basis for linguistic specification. Research Report RR-94-39, Deutsches Forschungszentrum für Künstliche Intelligenz, Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany. To appear in Machine Translation and the Lexicon, Springer, Berlin, 1995.

Krieger, Hans-Ulrich and Ulrich Schäfer. 1994. $\mathcal{TDL}$—a type description language for HPSG. part 1: Overview. Research Report RR-94-37, Deutsches Forschungszentrum für Künstliche Intelligenz, Saarbrücken, Germany.

Miller, George A., Richard Beckwith, Christiane Fellbaum, Derk Gross, and Katherine Miller. 1993. Introdution to wordnet: An on-line lexical database. Research report.

Ramalho, J. Carlos Ramalho, J. João Almeida, and Pedro Henriques. 1995. Algebraic specification of documents. In *AMiLP'95.* University of Twente.

Sperberg-McQueen, C.M. and Lou Burnard. 1994. *Guidelines for Electronic Text Encoding and Interchange (TEI P 3).* Chicago: ACH/ACL/ALLC.

Vilela, Mário. 1991. *Dicionário do Português Básico.* Edições Asa.

Vilela, Mário. 1994. *Estudos de Lexicologia do Português.* Almedina – Coimbra.

Vromans, Johan. 1995. Perl reference guide. Manual 5.000.3. Perl was designed by Larry Wall.

Wall, Larry and Randal L. Schwartz. 1991. *Programming PERL.* O'Reilly and Associates.