Bigorna - um conjunto de ferramentas para problemas de migração de grafia

José João Almeida
jj@di.uminho.pt

André Fernandes dos Santos
pg15973@alunos.uminho.pt

21 de Novembro de 2009

Resumo

As mudanças na grafia de uma língua causam vários problemas. Neste documento estudaremos algumas questões relativas ao Acordo Ortográfico da Língua Portuguesa e problemas relacionados com migrações.

Um conjunto de ferramentas (Bigorna) para conversão, classificação e comparação de diferentes versões de textos serão apresentadas, e vários exemplos serão discutidos.

Analizaremos também os problemas de (1) inferência de regras de tradução a partir de exemplos, e (2) a aplicação deste conjunto de ferramentas em novos problemas de mudança ortográfica.

Conteúdo

1 Introdução **3**

1.1 Contextualização 3

1.2 Motivação 3

1.3 Resumindo 3

1.4 Objectivos do Bigorna 4

2 Desenvolvimento do projecto **5**

2.1 Reunindo os recursos 5

2.2 Actualização do dicionário 5

2.3 Conversores 6

2.4 Classificador 8

2.5 Ferramentas lexicais 8

2.6 Bigorna-Kit 10

2.6.1 **bigorna-kit-base** 10

2.6.2 **bigorna-kit-full** 11

3 Trabalho futuro **11**

1 Introdução

1.1 Contextualização

A 16 de Dezembro de 1990 foi assinado em Lisboa o Acordo Ortográfico da Língua Portuguesa, por representantes de Angola, Cabo Verde, Guiné Bissau, Moçambique, São Tomé e Príncipe, Brasil, Portugal e uma delegação de observadores da Galiza. Este tratado tinha uma data de entrada em vigor prevista para 1994. O seu objectivo é unificar, de forma tão abrangente quanto possível, o vocabulário geral da língua portuguesa, e surgiu no seguimento de vários outros tratados falhados assinados ao longo do século XX.

Apesar de vários dos países mencionados terem já ratificado o tratado, a sua implementação global tem sido adiada. No entanto, Portugal, um dos principais responsáveis por esta demora, ratificou-o recentemente, e são esperados em breve novos desenvolvimentos neste processo.

A intenção de aproximar todas as variantes de português tem como consequência algumas alterações de forma, ou estrutura, e de conteúdo na própria língua. No que diz respeito às alterações de conteúdo, o critério fonético (ou da pronúncia) foi escolhido em detrimento do etimológico, o que levou a alguns casos de grafia múltipla, como *facto* e *fato*, ou *sumptuoso* ou *suntuoso*. Dificultando ainda mais a situação, parece haver alguma falta de consenso sobre determinadas palavras, e a ausência de um vocabulário ortográfico comum da língua portuguesa (previsto no Artigo 2° do Acordo) contribui para estas discordâncias.

1.2 Motivação

A implementação do novo Acordo Ortográfico obriga a uma revisão e actualização de várias ferramentas linguísticas já existentes e à criação de algumas novas, tarefas que se revelam mais complicadas se tivermos em conta os casos de grafia múltipla.

A criação de um conjunto de ferramentas capazes de comparar textos próximos e inferir regras de tradução seria uma boa ajuda para todo este processo, com a mais valia de poder servir para outros casos (p.e. adaptação de textos escritos em grafias mais antigas, ou actualizações linguísticas futuras).

O projecto Bigorna vem tentar dar resposta às dificuldades impostas pelas actualizações de grafia, desenvolvendo um *kit* de ferramentas capazes de gerar aplicações das quais os utilizadores se possam servir para facilitar a tarefa de migração.

1.3 Resumindo

Breve apanhado do cenário criado pelo Acordo Ortográfico:

- As alterações ortográficas ditadas pelo acordo não podem ser automaticamente determinadas, por assentarem em critérios fonéticos (por vezes

ambíguos);

- É importante a criação e manutenção de uma *Base de Conhecimento do Acordo Ortográfico* - `BCAO`, uma tabela contendo lemas, mudanças e regras, baseada em listas actualmente existentes, novos exemplos encontrados e os resultados do processamento de textos. Esta tabela estará num processo contínuo de actualização e evolução, sendo difícil saber com certeza quando ela se encontra completa;
- A questão essencial é como determinar as palavras candidatas a serem incluídas no `BCAO`. Tornou-se clara a necessidade de um projecto dedicado a este processo de migração, e assim foi criado o Bigorna.

1.4 Objectivos do Bigorna

Para auxiliar no processo de migração são necessárias várias ferramentas e recursos. Os objectivos inicialmente definidos foram:

- Estudo e levantamento dos recursos existentes ligados ao Acordo Ortográfico;
- Criação de um dicionário/corrector ortográfico para a nova versão da língua portuguesa;
- Construir um conversor de português pré-acordo para português pós-acordo;
- Concepção de um classificador (para identificar a variante usada num dado texto);
- Desenvolver as ferramentas anteriores de forma a possuírem o menor número possível de dependências;

Rapidamente nos apercebemos de que o conjunto de palavras alteradas era actualizado frequentemente de acordo com as novas submissões. Assim, decidimos:

- Centrar toda a informação numa única tabela - `BCAO`;
- Conceber um sistema que, dada essa tabela, gerasse:
  - um novo dicionário/corrector ortográfico;
  - um novo conversor;
  - um novo classificador;
- Desenvolver ferramentas para auxiliar na construção da `BCAO`:
  - concepção de um conjunto de ferramentas para encontrar candidatos à `BCAO` a partir de textos;
  - construção de uma aplicação para extracção de diferenças lexicais a partir de bi-textos;
- Tentar tornar as ferramentas anteriores tão genéricas e reusáveis quanto possível;

2 Desenvolvimento do projecto

2.1 Reunindo os recursos

Este projecto iniciou-se com uma procura de recursos relacionados com o acordo ortográfico, onde chegámos à conclusão que ainda existem poucos exemplos de textos em português europeu segundo a nova grafia (devido à abordagem mais activa que o Brasil tem tido relativamente ao acordo, textos escritos em português do Brasil actualizado são mais comuns).

Ainda assim, foram encontrados vários dicionários, correctores ortográficos e vocabulários *online*, bem como um número crescente de publicações sobre este tema. Dos vocabulários encontrados destaca-se uma *lista das palavras que mudam*, contendo mais de 9.000 entradas, publicada pelo Instituto de Linguística Teórica e Computacional (ILTEC) no seu Portal da Língua Portuguesa.

Desta listra extraímos as partes que nos interessavam para este projecto, obtendo uma tabela com a seguinte estrutura:

```
PT ::  BR ::  AO1990 ::  prefPT ::  prefBR ::  Com
```

onde `PT` e `BR` representam uma dada palavra na sua versão europeia e brasileira, `AO1990` a forma actualizada da palavra (ou as várias formas, se existir mais do que uma), `prefPT` e `prefBR` indicam as formas preferidas em cada país, e `Com` é um pequeno comentário que assinala quando esta situação não representa uma mudança relativamente ao passado, ou as formas que não devem ser usadas em cada país.

Descartando as linhas onde o comentário indicava não haver alteração, a lista ficou reduzida a 2.600 entradas. Segue-se um pequeno excerto:

```
ancilóide :: ancilóide :: anciloide :: anciloide :: anciloide ::
anciróide :: anciróide :: anciroide :: anciroide :: anciroide ::
androginóide :: androginóide :: androginoide :: androginoide :: androginoide ::
andróide :: andróide :: androide :: androide :: androide ::
anecóico :: anecóico :: anecoico :: anecoico :: anecoico ::
aneléctrico :: anelétrico, aneléctrico :: anelétrico, aneléctrico ::
     anelétrico :: aneléctrico,anelétrico :: aneléctrico nPT
```

Esta lista foi usada na construção da primeira versão da *Base de Conhecimento do Acordo Ortográfico*, `BCAO`.

2.2 Actualização do dicionário

Para a criação do dicionário actualizado decidimos usar o `jspell`, um corrector ortográfico e analizador morfológico livre. Existem vários dicionários para o `jspell`, entre os quais um de português europeu.

Além de ser *open source*, o `jspell` possui ainda a vantagem de permitir, através do Chuveiro de Dicionários (uma aplicação desenvolvida pelo projecto

Natura da Universidade do Minho) propagar as actualizações para dicionários de outros correctores ortográficos (`aspell`, `ispell`, `hunspell` e `myspell`), alguns dos quais usados por aplicações *open source* bem conhecidas como o Firefox ou o OpenOffice. Uma outra funcionalidade bastante útil no `jspell` é a capacidade de, partindo de um conjunto de lemas e as respectivas regras de derivação (para formação de plurais, conjugações verbais, etc) gerar uma lista exaustiva de palavras reconhecidas (o dicionário de português, por exemplo, tem pouco menos de 35.000 entradas, cujas formas flexionadas perfazem mais de 690.000 palavras).

A actualização do dicionário de português do `jspell` foi realizada procurando no dicionário palavras da `BCAO`, e substituindo-as pela sua nova forma. No caso de existirem várias formas possíveis foi escolhida a variante recomendada para o português de Portugal.

Este processo levou a cerca de 960 substituições, num total de 2.600 entradas na `BCAO` e 35.000 entradas no dicionário. Usando o `jspell` para gerar as derivações deste dicionário actualizado, com o comando

```
jspell -d portAO -e
```

é possível obter cerca de 11.500 palavras.

Foi produzido um pacote contendo os ficheiros, documentação e instruções de instalação. O dicionário foi actualizado de forma a manter a sua total compatibilidade com o `jspell`.

No futuro estudaremos a viabilidade da construção de um dicionário equivalente para português do Brasil. Está em curso a actualização dos dicionários de outros correctores ortográficos com recurso ao Chuveiro de Dicionários. Existe uma forte possibilidade de actualização de mais palavras com base nos resultados das ferramentas de análise léxica desenvolvidas no âmbito deste projecto.

2.3 Conversores

O segundo ponto da nossa ordem de trabalhos foi a criação de um conversor de textos da grafia pré-acordo para pós-acordo. Devido aos casos de grafia múltipla que, ainda por cima, está dependente da variante de Português a ser utilizada, não pôde ser criado um conversor único. Assim sendo, optámos por criar duas versões: uma capaz de actualizar português europeu, e a outra capaz de fazer o mesmo para português do Brasil (respectivamente `pt2ptao` e `br2brao`).

Mais uma vez, adoptámos como ponto de partida a `BCAO` e o `jspell` (tirando já partido do dicionário actualizado criado no ponto anterior). Já foi referida a capacidade do `jspell` de flexionar lemas. Vejamos alguns exemplos típicos de entradas num dicionário do `jspell`:

```
acalentar/#vt/XYPLD/
coiote/#nm/p/
laico/#a/fidp/
zinco/#nm//
```

Facilmente podemos comprovar que cada entrada tem a seguinte estrutura:

```
lema/#cat/reg
```

onde `lema` é um determinado lema, `cat` um identificador da sua categoria morfológica e `reg` um conjunto de caracteres que codificam as possíveis flexões de `lema`.

Da `BCAO` extraímos os lemas em português europeu e as suas formas actualizadas (`lema_pt/lema_ptao`), e do dicionário de português do `jspell` extraímos os mesmos lemas e as regras para a sua flexão (`lema_pt/reg`). Daqui gerámos duas listas paralelas `lema_pt/reg` e `lema_ptao/reg` que passadas, respectivamente, aos comandos `jspell -d port -e` e `jspell -d portAO -e` (expansão pelo `jspell` usando o dicionário de português pré-acordo e pós-acordo) resultaram na expansão paralela das versões pré e pós acordo das palavras da lista `BCAO`. Juntando-as obtivémos uma espécie de *hashtable*, em que cada entrada está estruturada da seguinte forma:

```
lema_pt/lema_ptao
```

Como o dicionário de português do `jspell` é específico para português europeu não tínhamos as regras de derivação dos termos de português do Brasil. Assim, decidimos usar as regras de cada termo em português europeu (presentes no dicionário) para flexionar o respectivo termo brasileiro e a sua forma actualizada. A partir da `BCAO` e do `jspell` foram criadas listas com a forma `lema_pt/reg/lema_br` e `lema_pt/reg/lema_brao`, depois agregadas sob a forma `lema_br/reg/lema_brao`. Por um processo semelhante ao relatado para a versão anterior obteve-se uma *hashtable* com a seguinte forma:

```
lema_br/lema_brao
```

O *script* que desenvolvemos, em Perl, para realizar a conversão foi escrito de modo a minimizar o número de dependências, para facilitar a sua instalação e utilização. Pela mesma razão, optámos por incluir a *hashtable* referida no próprio programa, resultando assim numa aplicação auto-contida, instalável em Linux a partir de uma *makefile*.

O programa faz a conversão varrendo o texto à procura de termos `lema_pt` e substituindo-os pelo respectivo `lema_ptao` (`lema_br` e `lema_brao` na versão brasileira). Foram incluídas algumas opções adicionais, como a preservação de *tags* XML e de comandos LaTeX.

De seguida mostram-se exemplos de execução de ambas as versões:

```
$ pt2ptao
A adopção do acordo implica a actualização de algumas ferramentas.
A adoção do acordo implica a atualização de algumas ferramentas.

$ br2brao
Ele fez um vôo rasante sobre a aréia.
Ele fez um voo rasante sobre a areia.
```

Os diferentes formatos de codificação de texto foram a fonte de várias dores de cabeças e alvo de um cuidado especial ao longo deste projecto. Ambos os conversores, bem como as outras aplicações aqui desenvolvidas, aceitam neste momento texto codificado nos formatos ISO-8859-1 (vulgo `latin1`) e UTF-8, sendo possível forçar a leitura do *input* segundo um destes formatos.

2.4 Classificador

Depois dos conversores, criámos um programa capaz de classificar um texto como estando escrito em português europeu ou do Brasil. Esta parte do projecto insere-se na categoria de trabalho extra, não tendo sido planeada inicialmente. No entanto constitui um complemento natural e facilmente desenvolvido a partir do material gerado anteriormente.

Este script, denominado `whichPT` inclui uma lista de palavras em português de Portugal, e outra em português do Brasil (ambas previamente geradas no desenvolvimento dos conversores). A classificação é efectuada somando o número de palavras que pertencem a cada lista encontradas num texto; no fim, considera-se que o texto está escrito na variante com o maior número de palavras presentes no texto.

Esta implementação permite facilmente expandir o *script* para mais variantes (por exemplo, português arcaico). Foram ainda acrescentadas opções para não contabilizar *tags* XML e comandos LaTeX, bem como diferentes níveis de verbosidade no *output* (desde imprimir o número de palavras encontradas para cada variante até imprimir cada palavra das listas encontrada no texto).

Testando este *script* com versões em português de Portugal e do Brasil do livro Amor de Perdição, de Camilo Castelo Branco, obtém-se o seguinte:

```
$ whichPT AmorPerd.ptPT AmorPerd.ptBR
AmorPerd.ptPT     pt
AmorPerd.ptBR     br
```

2.5 Ferramentas lexicais

Este ponto da ordem de trabalhos tem como objectivo a concepção de ferramentas capazes de, através da análise de textos, construir e manter aplicações para migração de grafias (como as que foram desenvolvidas nos pontos anteriores).

O primeiro passo passou pela criação de um *script* capaz de, a partir de duas versões de um texto, com grafias diferentes, detectar as mudanças de uma para a outra. Este *script*, denominado `lexdiff`, recebe como argumentos dois ficheiros, e começa por os pré-processar, eliminando pontuação e espaçamento excedentário. Depois, actua processando os resultados de um `diff` entre os dois ficheiros. Apresentam-se de seguida as primeiras linhas do resultado do `lexdiff` aplicado às versões portuguesa e brasileira anteriormente mencionadas do Amor de Perdição:

```
$ lexdiff -s -i AmorPerd.ptPT AmorPerd.ptBR
Castelo => CAS'TELO
António => Antônio
verão => vereão
Vila Real => Vila-Real
António => Antônio
```

```
académicos => acadêmicos
frequentava => freqüentava
```

Existem várias opções para controlar o comportamento do `lexdiff`. No exemplo anterior a flag `-s` serve para suprimir as ocorrências comuns (ou seja, mostra apenas os casos em que as duas versões diferem), e a opção `-i` torna a execução *case insensitive*. Também é possível imprimir o resultado agrupado e ordenado pelo número de ocorrências de cada caso:

```
$ lexdiff -ac -s -i AmorPerd.ptPT AmorPerd.ptBR
32 acadêmico => académico
16 Vila-Real => Vila Real
14 idéia => ideia
12 redargüiu => redarguiu
 7 gênio => génio
 6 cinqüenta => cinquenta
 5 Antônio => António
```

A partir de resultados como os observados em cima podemos encontrar mais palavras que possam ser incluidas na `BCAO`.

Para a inferência de regras seria de desejar um processamento mais fino, que permitisse observar *padrões de mudança*, independentemente da palavra em que ocorrem. Neste sentido foi implementada a possibilidade observar diferenças ao nível do caracter (`-m`). Se mostrarmos o contexto em que cada mudança aconteceu, e ordenarmos os resultados pelo número de ocorrências, obtemos algo como o seguinte:

```
$lexdiff -m -ac -ctx AmorPerd.ptPT AmorPerd.ptBR
36 et => ect
34 dêm => dém
18 déi => dei
17 güi => gui
15 qüe => que
 8 at => act
 7 eç => ecç
```

Este modo de processamento funciona através da construção de uma *confusion matrix* que relaciona cada caracter (e o seu contexto) com todos os caracteres (e contexto) em que se traduz. Apresenta-se de seguida um extracto da *confusion matrix* gerada no exemplo anterior:

```
'et'  => { 'ect' => 36 },
'dêm' => { 'dém' => 34 },
'déi' => { 'dei' => 18 },
'güi' => { 'gui' => 17 },
'qüe' => { 'que' => 15 },
'at'  => { 'aat' => 1,
           'apt' => 1,
           'act' => 8  },
'eç'  => { 'eaç' => 2,
           'ecç' => 7,
           'epç' => 2  },
```

A análise das *confusion matrix* geradas permitirá não só inferir regras de tradução, mas também encontrar as respectivas excepções.

Existem as mais variadas aplicações para o `lexdiff`:

- procura de palavras candidatas a serem incluídas na `BCAO`, para actualização do dicionário, corrector ortográfico, conversores e do classificador;
- actualização de textos escritos com grafia antiga (por exemplo, textos do projecto Gutenberg);
- avaliação do grau de semelhança entre dois textos, e quais as palavras ou padrões que mais mudam;
- deduzir regras de actualização (quando for regularmente detectado um padrão de mudança), bem como os seus desvios (quando uma excepção for sistemática);
- testar o funcionamento dos conversores desenvolvidos, ou desenvolvimento de outros conversores (entre quaisquer duas variantes próximas de uma língua).

Neste momento estamos a trabalhar na implementação de um modo de funcionamento que permitirá passar ao `lexdiff` uma lista de "padrões a vigiar", padrões esses que serão procurados durante a execução no sentido de ver que tipos de mudança sofrem. Está também em planeamento uma ferramenta complementar ao `lexdiff`, o `lexpatch`, que permitirá actualizar textos a partir de um ficheiro de alterações produzido pelo `lexdiff`.

2.6 Bigorna-Kit

Os resultados do desenvolvimento deste projecto podem dividir-se em dois grupos, de acordo com o perfil do seu utilizador-alvo:

a) ferramentas que auxiliam na migração imposta pelo Acordo Ortográfico - `bigorna-kit-base`

b) aplicações para construção de ferramentas de migração de grafia - `bigorna-kit-full`

Assim sendo, criámos dois *kits* Bigorna, cada um composto por várias ferramentas. Cada uma das ferramentas continua no entanto a poder ser instalada independentemente das outras.

2.6.1 bigorna-kit-base

Inclui os conversores e o classificador. Estes programas destinam-se a serem usados por utilizadores comuns, leigos na área da linguística e/ou programação, em tarefas como correcção ortográfica ou actualização de textos, e como tal, foram desenvolvidos de modo a funcionarem o mais independentemente possível de outras aplicações e ficheiros. Cada um destes programas consiste apenas num ficheiro de código que contém as listas de palavras necessárias e a documentação.

O dicionário `jspell` não foi incluído neste pacote (por requerer a instalação do `Lingua::Jspell`), o que poderá vir a acontecer em breve, quando disponibilizarmos instaladores automáticos que façam o cálculo das dependências (por exemplo, pacotes Debian, RPM ou CPAN).

2.6.2 bigorna-kit-full

Inclui, neste momento, todos os ficheiros usados na construção dos conversores e classificador, o dicionário `jspell` e o analizador de diferenças lexicais.

Permite que cada utilizador modifique os ficheiros a partir dos quais estas ferramentas são geradas, requerendo conhecimentos de programação e noções básicas de linguística. Destina-se a facilitar o acesso a este projecto a utilizadores envolvidos em processos de migração de grafias ou similares.

3 Trabalho futuro

Apesar de o período do Sapo Summerbits ter chegado ao fim, este projecto continuará a ser desenvolvido.

Está neste momento em curso a replicação do dicionário de português actualizado do `jspell` para outros correctores ortográficos, recorrendo ao Chuveiro de Dicionários, o que permitirá alargar o leque de potenciais utilizadores. Dependendo dos resultados obtidos com as ferramentas lexicais, poderá vir a encontrar-se um conjunto de palavras ainda não incluídas no dicionário. Está também a ser estudada a hipótese da criação de uma versão para português do Brasil deste dicionário.

O campo das ferramentas lexicais é o que promete resultados mais férteis. O `lexdiff` continua a ser melhorado, e servirá de base a grande parte do trabalho subsequente: aperfeiçoamento dos conversores construídos (acrescentando palavras ou mesmo regras de conversão), desenvolvimento de outros conversores (por exemplo, para conversão de grafia de textos do projecto Gutenberg para português corrente), construção de listas de palavras alteradas mais extensas e avaliação da qualidade de todas estas ferramentas.

As várias aplicações foram desenvolvidas de forma a serem tão independentes quanto possível, quer da plataforma, quer de outras aplicações. Consideramos que o conjunto de todas as aplicações desenvolvidas dá resposta a uma boa parte das necessidades que se criaram com o Acordo Ortográfico, e pretendemos também desenvolver versões para `Windows` (o sistema operativo `MacOS X` é já parcialmente suportado).

Continuaremos a tentar recolher textos com grafias diferentes; à medida que mais países ratificam e implementam o acordo, é natural que comecem a surgir publicamente mais textos sob a nova grafia. Existe também ainda a hipótese de procurar textos antigos cuja grafia já tenha entretanto sido traduzida manualmente. Ambos os tipos podem vir a servir para *benchmarking*, aferição e discussão de casos.