Processing and extracting data from an open dictionary of the Portuguese language

Alberto Simões[1], José João Almeida[2], Rita Farinha[3]

[1] Escola Superior de Estudos Industriais e de Gestão, Instituto Politécnico do Porto, Portugal
[2] Departamento de Informática, Universidade do Minho, Portugal
[3] Project Gutenberg, Distributed Proof-readers, Portugal
alberto.simoes@eu.ipp.pt, jj@di.uminho.pt, rfarinhadp@gmail.com

Abstract

Synonyms dictionaries are useful resources for natural language processing. Unfortunately their availability in digital format is limited, as publishing companies do not release their dictionaries in open digital formats. Dicionário-Aberto (Simões and Farinha, 2010) is an open and free digital synonyms dictionary for the Portuguese language. It is under public domain and in textual digital format, which makes it usable for any task.

Synonyms dictionaries are commonly used for the extraction of relations between words, the construction of complex structures like ontologies or thesaurus (comparable to WordNet (Miller et al., 1990)), or just the extraction of lists of words of specific type.

This article will present Dicionário-Aberto, discussing how it was created, its main characteristics, the type of information present on it and the formats in which it is available. Follows the description of an API designed specifically to help Dicionário-Aberto processing without the need to tackle with the dictionary format. Finally, we will analyze the results on some data extraction experiments, extracting lists of words from a specific class, and extracting relationships between words.

1. Introduction

Dicionário-Aberto[1] pretends to be an open and free dictionary for the Portuguese language. To an extent, the main idea can be compared to Wiktionary[2] with some differences (namely adding a review process to contributed entries).

At the present time the project is still in its beginnings, incorporating a base lexicon. As the creation of a full dictionary is colossal and given the lack of funds for this project, the authors decided to use an old dictionary from *Cândido de Figueiredo*, available in paper format at the Portuguese National Library. This dictionary age (it was published in 1913) puts it under public domain which gives us full permission to use the document for any objective.

Previous work (Simões and Farinha, 2010) presents the details on the paper dictionary, its problems (the more relevant problems are the fact of being written in Old Portuguese, and the related problem of being out-of-date on some actual words/definitions), how its transcription was performed, and what methods are being used to convert the dictionary into different formats (XML, StarDict, PDF and others).

At the present moment the transcription is complete and the text incorporation in the web site is being finished[3]. This process took about three years at a constant rate of 200 words added per day. Until the full incorporation is finished Dicionário-Aberto language will not be modernized (there are already some studies performed to make that tasks semi-automatic) and it will not be opened to community contribution.

Nevertheless, some tools to process the dictionary are already available and interesting results can be already extracted from it. As these methods are dependent on the document format but not highly dependent on the language being used, they can be re-applied after the language modernization take place.

In this article we will focus on this problem: how to use Dicionário-Aberto to extract words relationships or word lists, to be used as semantic structures for natural language processing tasks.

We first summarize the format used to encode Dicionário-Aberto (section 2.). In section 3. we discuss the API (Application Programmers Interface) that was prepared to help and systematize the dictionary processing task. Section 4. shows some experiments performed extracting different kinds of information from Dicionário-Aberto together with some basic evaluation. We conclude on section 5. with some insights of Dicionário-Aberto future.

2. Current Dictionary Pipeline

Dicionário-Aberto's project pipeline begins with the paper dictionary digitalization, followed by the optical character recognition and transcription/validation process. This process is assisted with different kinds of quality assurance tests from syntax to completeness checks. This first part of the process results in a textual format version of the dictionary using a "Project Gutenberg"-like syntax (Foundation, 2009). This syntax is simple but ambiguous. As an example, it does not contain clear distinction on definition senses.

To enrich and make the dictionary processing easier, a conversion process was developed. It rewrites this basic syntax into an XML format, using simple heuristics and specific word lists. The chosen XML format is a subset of TEI (Text Encoding Initiative (Vanhoutte, 2004)) schema for dictionaries encoding. An example of a dictionary entry in TEI format is presented in figure 1. More TEI structure will be

[1] Available from http://dicionario-aberto.net/
[2] http://www.wiktionary.org/
[3] The geographic and onomastic appendixes are being imported.

added to the dictionary in the future.

```
<entry>
 <form>
  <orth>Cachimbo</orth>
 </form>
 <sense>
  <gramGrp>m.</gramGrp>
  <def>
   Apparelho de fumador, composto de um...
   Peça de ferro, em que entra o espigão...
   Buraco, em que se encaixa a vela do...
  </def>
 </sense>

 <sense>
  <usg type="geo">Bras. de Pernambuco.</usg>
  <def>
   Bebida, preparada com aguardente e mel.
  </def>
 </sense>

 <sense>
  <gramGrp>Pl.</gramGrp>
  <usg type="lang">Gír </usg>
  <def>Pés.</def>
 </sense>

 <etym ori="químb">(Do químb. quixima)</etym>
</entry>
```

Figure 1: Dicionário-Aberto entry encoded in TEI.

For better understanding of the following sections it is important to summarize the structure of Dicionário-Aberto encoded in TEI.

- The dictionary is a sequence of entries:

  $$Dic = Entry^{\star}$$

  Each entry, as shown in figure 1, contains information about one word, including different senses.

  In some cases lexicographers prefer to separate senses in different entries (especially when they have different grammatical information) and in some other contexts prefer to include them on the same entry. This discussion is not important in this context as TEI can encode both situations. Dicionário-Aberto's lexycographer was responsible for those choices when producing the original dictionary.

- Each entry is a triple, including the word form, a sequence of senses definitions and optional etymology information:

  $$Entry = WordForm \times Sense^{\star} \times Etymology?$$

- Word form includes only one mandatory value: one orthographic word form. It might also include other orthographic forms and full or partial phonetic transcriptions:

  $$WordForm = OrthcForm^{\star} \times PhoneticForm^{\star}$$

- Word senses have two main fields: the word grammatical information (or part-of-speech) and the sense definition.

  Note that a sense might include different but similar descriptions, that are being separated by a new line. TEI is not clear if these different descriptions should be placed in different senses or kept as a single one. Our approach follows simplicity, putting them as part of the same sense, but making it possible to split them in the future.

  Word senses might also include some other data, like domain or geographic usage information.

  $$Sense = GramInfo \times Definition \times Usage\text{-set}$$

  Grammatical information is a non-structured string using a closed set of abbreviations[4].

- Finally, the etymology field at the present moment is just one non-structured string. In future Dicionário-Aberto versions it should be bettered.

The full dictionary in XML format takes about 30MB of disk space, with more than 128 000 entries and 152 000 senses. Processing this XML file using a DOM (Document Object Model) approach is heavy, given the complexity of the XML format.
Therefore, with efficiency in mind, the XML file was spliced in different chunks, one for each entry, that were inserted in a relational database.
This database has a compound key with the orthographic form of the word and its homograph number (or 1 if no homograph is present). A column follows with the entry encoded in TEI and a final column contains a normalized version of the term.
The processing API described in the next section uses this database instead of processing the XML file.

3. Systematic Dictionary Programming API

The main structure of the TEI file is a list of entries. Therefore, the most convenient way to process this structure is cycling thought all the entries.
When looking at an entry, we want to have access to the orthographic forms and to the usage information, but if we are trying to extract semantic relationships, then we will want to access the term definition, that is, to the different senses.
These two kind of approach are the most common on most data extraction applications. Therefore, our API is mainly composed by two methods, one prepared to cycle through entries and the other to cycle through senses.

3.1. Processing Dictionary Entries

There is a common procedure when searching a dictionary: cycle through all dictionaries entries, looking for the word and deciding if that word is, or not, relevant towards our main objectives. This approach is both true when talking

[4]Tests were run to force coherence on the used abbreviations.

about real world dictionaries and when talking about processing automatically a digital dictionary.
Therefore, the provided `process_entries` API method has that same behavior: it looks at every word in the dictionary (by default in alphabetic order) and lets the user specify a function to process its contents.
In the same way that when searching a paper dictionary we look at the word and decide whether to read or not its definition, this method also provides such a mechanism. Along with the code that will process the dictionary entry the user may supply a set of restrictions about the word. This restriction can specify a full word, specify a prefix or suffix, or just some characters that should be present in the word.
In summary, the `process_entries` (PE) method[5] can be defined as:

$$\text{PE} : \mathit{restriction}\text{-set}, (\mathit{word} \times \mathit{entry} \rightarrow \mathbf{any}) \longrightarrow \mathbf{any}^{\star}$$

Each word can be processed or skipped accordingly with the process objectives, checking the word (as described above) or looking into the morphological categories, domain or geographic information.
The Dicionário-Aberto API is defined in Perl. While it might be difficult to read for non-Perl programmers we defend that presenting some code will help to understand how the API works.

```
$DA->process_entries(
   {
      grep_words => qr/ch/,
      words_like => 'p%',
   },
   sub {
      my ($word, $entry) = @_;
      ...
   }
)
```

By default, this method applies the supplied function to all entries in the dictionary, passing to this function the word and the full TEI entry (valid XML fragment).
The method allows the user to filter the entries to be processed. For efficiency purposes it supports two filtering mechanisms: `grep_words` restricts the process to words matching a Perl regular expression (slower); `words_like` restricts the process to words matching an SQL-like statement[6] (faster).

3.2. Processing Word Senses

The second approach to process the dictionary is inspecting each word sense and processing it independently. This method is especially useful when the application is searching for a special morphologic property or a special usage method.

[5]In this and the following method signatures the dictionary will not be represented because of the scarce line space and because being a method it is implied that it will be applied to an object, in this case, the dictionary.

[6]It is relevant to have both filtering mechanisms given that SQL-like is faster than Perl regular expressions, but Perl regular expressions are more expressive.

Therefore, the `process_senses` method allows the same filter mechanisms as the `process_entries` method and adds options to filter morphological properties — `grep_morph` — and term usage information (geographic or domain) — `grep_usg`.
We can specify this method (PS) signature as follows:

$$\text{PS} : \mathit{restriction}\text{-set}, \mathit{proFunction} \longrightarrow \mathbf{any}^{\star}$$

where

$$\mathit{procFunction} = (\mathit{word} \times \mathit{morph} \times \mathit{def} \times \mathit{usg} \rightarrow \mathbf{any})$$

As can be noted in the signature, the user function will not process the full entry in TEI format, but the word, the morphological information, the definition (note that it might contain XML tags) and an associative array with usage information (if any).

```
$DA->process_senses(
   {
      ...,
      grep_morph => qr/adj/,
      grep_usg   => { geo => qr/bras/ },
   },
   sub {
      my ($word, $morph, $def, %usg) = @_;
      ...
   }
)
```

4. Dicionário-Aberto Data Extraction

The common data extraction task performed over a dictionary is the extraction of relationships for the creation of semantic structures, like thesauri or ontologies(Oliveira et al., 2008; Zesch et al., 2008).
In this article we will also extract similar relations from Dicionário-Aberto, but our main objective is not to defend our data extraction approach, but to defend the usefulness of Dicionário-Aberto for natural language processing and the easiness of this task when using the presented API.
This section includes two case studies. The first is devoted to the extraction of word lists from specific word classes, like animals or plants. The second focuses on the extraction of word relationships.

4.1. Extracting Word Lists

Lexicographers write definitions with a strict structure. If a word has a sense explaining it is the name of a plant, then the definition starts with[7] "*plant that...*", and if it is the name of a fish, then the definition starts with "*fish that...*"
While this structure is maintained in most cases, we need to have in mind that Dicionário-Aberto is one hundred years old. Therefore, descriptions were written manually and maintaining coherence between entries was a difficult task. Nevertheless, we did some experiments for the extraction of word lists based on this assumption, creating lists of quadrupeds, fishes, birds, insects, generic animals, plants and trees. Note that the class *animals* include some of the other classes.

[7]In these examples we will be using a rough translation of the dictionary entry to make it easier for non-Portuguese readers to understand our explanations. Full examples will be presented in Portuguese.

```
my $DA = new DicionarioAberto;

$DA->process_senses(
   sub {
      my ($word, $morph, $def) = @_;

      if ($def =~ /^quadrúpede/i) {
          print QUADRUPEDES "$word\n";
          print ANIMAIS     "$word\n";
      }
      if ($def =~ /^(ave|pássaro)\b/i) {
          print AVES    "$word\n";
          print ANIMAIS "$word\n";
      }
      if ($def =~ /^peixe\b/i) {
          print PEIXES  "$word\n";
          print ANIMAIS "$word\n";
      }
      if ($def =~ /^insecto\b/i) {
          print INSECTOS  "$word\n";
          print ANIMAIS   "$word\n";
      }
      if ($def =~ /^animal\b/i) {
          print ANIMAIS "$word\n";
      }
      if ($def =~ /^planta\b/i) {
          print PLANTAS "$word\n";
      }
      if ($def =~ /^árvore\b/i) {
          print ARVORES "$word\n";
      }
});
```

Figure 2: Script to extract word lists

A simple extraction tool can be developed with a few lines of code as shown in figure 2[8]. It runs in about 1 minute and 9 seconds on a 2.4GHz Core 2 Duo running Mac OS X. Table 1 quantifies the number of words extracted for each class.

class	hits	precision
quadrupeds	47	98%
fishes	459	96%
birds	742	98%
insects	265	100%
animals	1660	91%
plants	2307	100%
trees	1352	100%

Table 1: Word list sizes

The same table also presents an evaluation on the results. The third column includes the precision on a hand evaluation task on 100 random elements from the list. Just the first class (quadrupeds) was evaluated on its totality.
For instance, on the quadrupeds list, the only false positive is "*bêsta*" that can be seen as a synonym for quadrupeds, but not an instance. The same happens with fishes and birds, where diminutive or augmentative terms are found.
The word precision drops in the animals class, mainly because the rule searches for "*animal*" at the beginning of the definition,while these definitions (from the set analyzed by hand) are, mostly, related to animal classification terms (like "*herbivore*").

4.2. Extracting Word Relationships

This section presents a more complex method for extracting data from a definitions dictionary, that has been used by other researchers. The idea is similar to the one presented above, but the patterns are more complex.
The first step consists on detecting and analyzing the patterns that should be used for the extraction task.
Instead of defining ahead the set of relations to be extracted, based on the common relations used on thesaurus or ontologies, we decided to analyze dictionary word sequences[9] (n-grams), sort and count them, and find out the most commonly used patterns.
Each pattern was associated to a relation or discarded if no interesting knowledge could be retrieved from it.
Table 2 shows a few of the most occurring and interesting patterns found, together with the inferred relation[10].

pattern	relation
o mesmo que	synonym
o mesmo ou melhor que	synonym
acto ou effeito de	effect/cause (noun/verb)
acto de	effect/cause (noun/verb)
relativo á|a|ao <NOUN>	adjective/noun relation
gênero de	hypernym/hyponym
espécie de	hypernym/hyponym
aquillo que	subject/action
aquelle que	subject/action
que não é <ADJ>	antonyms

Table 2: Some relevant patterns to extract word relationships.

The extraction tool, written with the defined API, it not much more complex than the one presented before, as can be seen on figure3
This example takes about 1 minute and 7 seconds running and extracting a total of 34 308 relations, divided in 16 038 synonyms, 2 292 cause/effect, 5 361 noun/adjective, 6 992 hypernyms/hyponyms, 3 147 verb/action and 478 antonyms. These values can be raised if the patterns are not searched in the beginning of the definition but anywhere in the text. However, this will result in a fewer precision.
Table 3 shows some relations extracted using this tool. An evaluation similar to the performed in the previous example was also conducted for these results. Main problems found were not related to wrong relations but to incomplete

[8]Note that the code shown in this article is not supposed to work by itself, as some details were removed for legibility and compactness.

[9]This extraction task was also performed using the defined API in a simple processor that will not be shown here.

[10]For non English readers follows a rough translation for each line on the table: *the same*; *the same or better than*; *act or effect of*; *act of*; *relative to*; *type of*; *specie of*; *the thing that*; *the one that*; *that is not*.

```
my $DA = new DicionarioAberto;
my $word = qr/([^_,.;:]+)/;

$DA->process_senses(
  sub {
    my ($word, $morph, $def) = @_;
    $_ = $def;

    if (/^o mesmo (ou melhor)?que $word/) {
        print "SYN: $word | $1\n";
    }
    if (/^acto ou effeito $word/) {
        print "EFFECT: $word | $1\n";
    }
    if (/^relativo (á|ao|a) $word/) {
        print "ADJ: $word | $2\n";
    }
    if (/^(gênero|espécie) de $word/) {
        print "HYP: $word | $2\n";
    }
    if (/^(aquillo|aquelle) que $word/) {
        print "ACTION: $word | $1\n";
    }
    if (/^que não $word/) {
        print "ANTON: $word | $1\n";
    }
});
```

Figure 3: Word relationship extraction code.

words, as this simple approach does not have any syntactical help to find out if the word being related is a single word, multi-word term, or other.
From the 100 word lists analyzed:

- just one relation from the cause/effect list was evaluated as wrong: "*resaudação/saudação*" because the definition in the dictionary is not precise/complete[11].
- three antonym entries are wrong given the lack of syntax knowledge on the extraction tool: "*ileso/não é ou não está leso*" "*estrangeiro/o é do país em que está*" "*impune/não é ou não foi punido*". In these examples underlined words are the extracted ones.
- remaining lists were considered correct.

5. Conclusions

Dicionário-Aberto shown to be a valuable resource for the extraction of knowledge for the Portuguese language. The fact that it is written in old Portuguese can be limiting at first, but with the proper modernization tool that can be minimized. Current experiments show that the dictionary can be modernized to 90% automatically. This value is pretty promising if we take into account the diversity of words present (the dictionary includes more than 128 000 entries).

[11]If while evaluating we only consider the dictionary itself and not the common knowledge of the language, then all evaluated cause/effect relations were correct.

relation	words	
synonyms	*trabalhucar* *trafulha* *pélvis* *mijo*	*trabalhar* *trapaça* *pelve* *urina*
effect/cause	*união* *remoção* *picada*	*unir* *remover* *picar*
adjective/noun	*topográfico* *toireiro* *polaco*	*topografia* *toiro* *Polônia*
hyponym/hypernym	*taipoca* *selenato* *elefantina*	*árvore* *mineral* *tartaruga*
subject/verb	*tinhoso* *surdo* *santidade*	*sofre tinha* *não ouve* *é santo*
antonyms	*raro* *touro* *material* *claro* *canhoto*	*denso* *castrado* *espiritual* *escuro* *destro*

Table 3: Extracted word relationship examples.

The availability of an API to process the dictionary reduces drastically the time necessary for the tools development, making it easy to filter entries accordingly with the words, morphologic or usage information, and center the developer attention to its main objective.
While the presented experiments are naive they illustrate both the simplicity of the API use and the relevance and value of Dicionário-Aberto as a source for lexical and semantic knowledge.

6. References

Project Gutenberg Literary Archive Foundation. 2009. Project gutenberg. `http://www.gutenberg.org/`.

George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine Miller. 1990. WordNet: an on-line lexical database. *International Journal of Lexicography*, 3:235–244.

Hugo Gonçalo Oliveira, Diana Santos, Paulo Gomes, and Nuno Seco. 2008. PAPEL: A dictionary-based lexical ontology for portuguese. In *PROPOR'08: Proceedings of the 8th international conference on Computational Processing of the Portuguese Language*, pages 31–40, Berlin, Heidelberg. Springer-Verlag.

Alberto Simões and Rita Farinha. 2010. Dicionário aberto: Um novo recurso para PLN. *Vice-Versa*, (16), April. forthcomming.

Edward Vanhoutte. 2004. An Introduction to the TEI and the TEI Consortium. *Lit Linguist Computing*, 19(1):9–16.

Torsten Zesch, Christof Müller, and Iryna Gurevych. 2008. Extracting lexical semantic knowledge from Wikipedia and Wiktionary. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Bente Maegaard,

Joseph Mariani, Jan Odjik, Stelios Piperidis, and Daniel Tapias, editors, *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, may. European Language Resources Association (ELRA). http://www.lrec-conf.org/proceedings/lrec2008/.