

DRAFT: Orthoepikon: a toolbox to build reading-aloud assistants

Carmen Arronis i Llopis¹,
Mikel L. Forcada², Sergio Ortiz-Rojas², Carlos Pérez Sancho²,

¹Departament de Filologia Catalana
Universitat d'Alacant, E-03071 Alacant

²Departament de Llenguatges i Sistemes Informàtics
Universitat d'Alacant, E-03071 Alacant, Spain

April 25, 2006

Abstract

Orthoepikon is a toolbox which allows for the automatic construction of systems to assist people to read texts aloud correctly. Each system is generated from a linguistic data file (containing a dictionary and rules) written in a standard format based on XML. The systems read plain text, HTML or RTF, and add to them annotations which are easy to interpret and indicate the correct pronunciation in difficult or exceptional cases. Annotation is almost instantaneous thanks to the use of finite-state techniques. The application of the described tools to a particular language is briefly described.

1 Introduction

This document presents the linguistic and computational design of Orthoepikon, a toolbox to create programs that will assist media speakers and, more generally, any person to read written texts aloud with a correct pronunciation. In particular, it will assist readers when the orthography of the text is not sufficient or

tends to be misinterpreted due to lack of fluency or to the influence of other languages, or when the accent of the speaker deviates from a certain standard that should be observed. The programs generated by Orthoepikon add to the text simple annotations which indicate the correct pronunciation in difficult cases.

Programs generated by these tools run on regular desktop or laptop personal computers and automatically add to the text simple marks that indicate the correct pronunciation in difficult cases. These tools are better suited to languages having orthographical systems which are approximately phonetic (Spanish, Portuguese, Catalan, Basque, Italian, etc.) than to languages having more complex letter-sound relationships such as English.

Annotations may be necessary due to a variety of causes; among them,

- because of the residual ambiguity of a writing system which is basically phonetic;
- because of occasional divergences between the standard orthography of a language and a particular dialectal pronunciation which is desired;

- because of the simplified pronunciation of certain consonant groups of classical origin (such as initial *ps-* or *mn-*);
- to avoid incorrect pronunciations, for instance, due to the influence of another language (reading in Spanish **carácteres* [ka'rakteres] where it clearly reads *caracteres* [karak'teres] or **espectroscopía* [espektrosko'pia] where it says *espectroscopia*) [espektros'kopja] even when the orthography is clear;
- to correctly pronounce words having orthographical exceptions (such as the Spanish words *Texas* and *México*, which have to be pronounced as if they were written with a *j*: ['texas], ['mexiko]) or foreign proper names.
- to annotate pronunciation changes due to the contact with other words.

These marks or annotations have to be designed in such a way that

- they should be easy to learn, easy to identify and easy to interpret at the usual reading speeds after a brief learning phase¹
- they should be minimal, that is, they should annotate the text only where it is necessary;
- they should preserve the original orthography of the text (especially important in second-language learning situations);
- they should be rapidly generated so that there is virtually no delay between the generation (authoring) of the text and the time of reading it aloud; and

¹Therefore, they should not be based on phonetic alphabets such as the International Phonetic Alphabet, known only to specialists, but should be whenever possible based instead on the orthography of the language itself.

しんかんせん ← *ruby text*
 新幹線 ← *ruby base*

Figure 1: *Ruby* or *Furigana* annotation of the three ideograms (or *kanji*) corresponding to the Japanese expression *bullet train* with symbols from the Japanese syllabic alphabet *hiragana* to indicate its pronunciation, *shinkansen*: *shi-n*, *ka-n*, and *se-n*, respectively. Figure taken from <http://www.w3.org/TR/ruby/>.

- should be easily generated using a system which may be accessed from any computer connected to the net or easily installed in an ordinary desktop or laptop personal computer.

As will be shown below, one of the objectives of Orthoepikon is that the pronunciation information generated by the system is independent of the particular presentation that it will have. This will make it easy to present it in more than one possible way, and has allowed us to design the phonetic annotation component independently of the presentation component. This is important, for example, to address ergonomical issues or readers' preferences. Annotations follow the style of *ruby* or *furigana* annotations used in Japanese to indicate the correct pronunciation of ideograms (*kanji*): they are symbols in a smaller type which are placed on top of symbols needing annotation, as may be seen in figure 1.

It is also possible to use superscripts as in the following examples from French, which show examples of *liaison* and some exceptional pronunciations:

Un grand^t homme, un long^k hiver, les^z
 enfants, neuf^v ans, une fe^amme, un
 mon^esieur, de sec^gonde main.

The systems generated are based on dictionaries and pronunciation rules, for words in

isolation as well as for words in contact (phonetic phenomena that occur at the beginning or the end of words, sometimes called *external sandhi*, *liaison*, etc.). Dictionaries may also contain multi-word units, for instance, to decide in some simple cases the pronunciation of homographs words (having the same orthography) which are heterophones (have a different sound), such as in the case of the English word *tears* which may sound [ˈtɪə(r)z] (as in *salty tears*) or [ˈteə(r)z] (as in *tears apart*).

Compilers transform these linguistic data in finite-state representations which speed up the search for words in the pronunciation dictionary. As to the treatment of the different text formats, the resulting systems may read plain texts (ASCII, ISO-8859-1) and texts formatted using character-based markup schemas (RTF, HTML). In the case of HTML, the speed of the annotator² allows for the annotation on the fly of Internet texts accessed through a browser.

The following sections describe: the annotation engine (section 2), the nature and format of the linguistic data used to define these systems (section 3), and the compilers (section 4). Section 5 describes a practical application of these tools. Concluding remarks are given in section 6.

2 The annotation engine

The annotation engine has four modules forming a text pipeline³ or assembly line, as shown in figure 2:

- a *deformatter* separates text to be annotated from the format information in the original document
- a *preannotator* performs temporary (context-dependent) annotations and

²tens of thousands of words a second in a regular desktop computer

³In which the input of module n is the output of module $n - 1$.

definitive (context-independent) annotations showing the pronunciation of each word (to perform this task, the preannotator reads in a finite-state machine (FSM) resulting from the compilation of the annotation dictionary);

- a *postannotator* confirms or discards the temporary annotations according to context, and
- a *reformatter* restores the original format to the text and produces a graphical presentation of the pronunciation annotations. In the case of HTML documents which are being annotated during navigation, the deformatter may transform links so that they point at the web where the annotation system resides so that clicking on these links causes the browser to deliver the annotated version of the document pointed at by the original link.

The text flow between the modules of the system is based on XML. The deformatter encapsulates format information in the original text in `<![CDATA[...]]>` sections which are not taken into account by the preannotator and the postannotator (for instance, a label such as `` would be sent as `<![CDATA[]]>`). When the annotation process ends, the deformatter eliminates the `<![CDATA[and the]]>`. Most of the words in the text are transferred intact, except when they receive an annotation (either temporary or definitive). Annotations are included in words as XML entities, so that the annotation process is separated from the presentation process (which is the task of the reformatter). In this way, it is possible to decide which graphical form will be used for annotation. For instance, the French word *seconde*, in which the *c* sounds [g] instead of [k] will be sent by the preannotator as `se&c_g;onde`.

The deformatter and the reformatter are open-sourced versions of those used in machine translation project `interNOSTRUM` [Canals-Marote et al., 2001a, Canals-Marote et al., 2001b], <http://interNOSTRUM.com>. The preannotator uses program `lt-proc` in the lexical processing package `lttoolbox`, which is part of the `apertium` shallow-transfer machine translation system.⁴ The postannotator is completely generated by the corresponding compiler (see section 4). All of the code in `Orthoepikon` has a GPL license⁵ or is based in code having the GPL license.

All of the modules are based on finite-state technology. In particular, the preannotator reads the necessary information as a letter transducer —a kind of finite-state transducer [Roche and Schabes, 1997]— which is generated by a compiler (program `lt-comp` in module `lttoolbox` from the preannotation dictionary; the remaining three modules are based on lexical processors generated using `lex` [Lesk, 1975] as an intermediate representation.

3 Linguistic data

The preannotator and postannotator modules are generated, using compilers, from an XML linguistic data file. Figure 3 shows the DTD (*document type definition*) specifying this format. The DTD may be used to assist who has to validate, create or modify linguistic data for `Orthoepikon`. All data go inside the `orthoepikon` element, which consists of a definition of the alphabet (`alphabet`), an annotation definition section (`annotDefSec`), a preannotation section (`preannotSec`) and a postannotation section (`postannotSec`).

⁴See [Corbí-Bellot et al., 2005] and <http://www.apertium.org> for updates.

⁵GNU General Public License, <http://www.gnu.org/copyleft/gpl.html>.

Annotations defined in `annotdefSec` are grouped in two sections: definitive (`daDefSec`) and temporary (`taDefSec`), each of which defines, respectively, zero or more definitive annotations (`daDef`) and zero or more temporary annotations (`taDef`). Each definitive annotation has three attributes: its name (`n`), the orthographical form which is annotated (`orth`) and the annotation that will appear on top of the orthographical symbols (`annot`). Each temporary annotation has two attributes: its name (`n`) and the orthographical form affected (`orth`), and contains one or more annotation definitions (`aDef`), each one with its form (`n`), because these annotations may be resolved differently depending on the context.

The preannotation section (`preannotSec`) has two parts: the *paradigm* definition section (`pDefSec`) and the dictionary itself (`dic`). Each paradigm is defined in a `pDef`, with a mandatory name `n` and an optional indication of the lemma (`lemma`), and is made of one or more entries `e` that may appear in the same context. Each entry `e` is built by concatenating, in any order, zero or more annotated text strings `t` and zero or more paradigm invocations `p`, each one of which is referred to by its name `n`. The annotated text is a concatenation of text (`#PCDATA`), definitive annotations `da`, and temporary annotations `ta`, invoked by their name `annot`, in any order. The dictionary `dic` is a set of one or more entries `e` such as the ones defined above, one for each word.

The postannotation section may also have a paradigm definition section `pDefSec` and zero or more context sets (`contexts`) to solve temporary annotations. Each `contexts` element contains one or more annotations `annot`, each one with its name `n`, indicating the definitive form the temporary annotation will take. Annotations `annot` contain one or more contexts (`context`) in which the annotation (represented by the empty element `annotation`) takes the form indicated by `annot`, or that indicated by

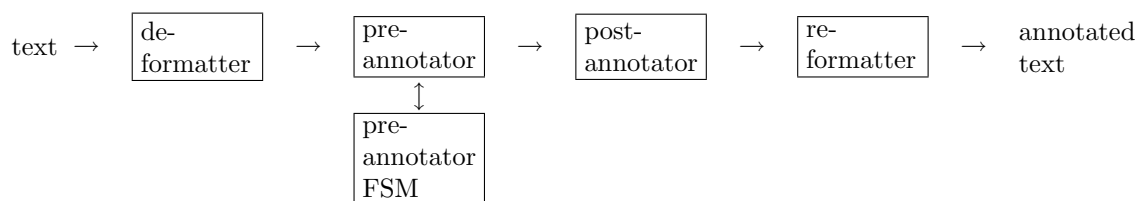


Figure 2: The modules in the Orthoepikon annotation system

default, that is, the form it will take if none of the specified contexts occurs. The contexts may have a left part a right part made up of zero or more paradigms (p), annotated text segments (t), or whitespace (ws).

Figures 4 and 5 show an example of a linguistic data file with some rules for French.

4 The compilers

There are two compilers which read the same linguistic data file:

- The preannotator compiler (program `lt-comp` in package `lttoolbox` of project Apertium with option `-c`) reads a linguistic data file and generates a finite-state transducer that will be read by a generic pre-annotation module (program `lt-proc` in package `lttoolbox` with option `-c`). This transducer reads the words in the text to be annotated and writes the annotated words.
- The postannotator “compiler” —really an XSL⁶ stylesheet— reads the data file and generates a `lex` program that is compiled into a specific postannotation module.

⁶Extensible Stylesheet Language, <http://www.w3.org/Style/XSL/>.

5 A practical case

The tools described (programs, compilers, DTD) have been used to create an annotation system for the Valencian variety of Catalan, commissioned and funded by the Acadèmia Valenciana de la Llengua (Valencian Academy of Language).

The system is accessible through the net both through the Acadèmia’s web (<http://www.avl.gva.es>) or directly through <http://sao.dlsi.ua.es>; it may also be installed as a Linux or Windows binary. It is based on a 23.000-lemma dictionary,⁷ performs 72 different annotations and uses a dozen context rules to decide on the the temporary annotations. It also contains some multiword units which represent common word sequences and permit the correct annotation of homographs which are heterophones such as *set* ([ˈsɛt] *seven*, [ˈsɛt] *thirst*, *tennis set*) or *deu* ([ˈdɛu] *ten*, [ˈdɛu] *he/she/it owes*).

In a preliminary evaluation on news releases from Europa Press with the above data, the percentage of words that the system should have processed and have indeed been correctly annotated is 98.9%. To improve this result, it would be necessary to dramatically increase the amount of entries in the linguistic data file. In subjective tests with learners of Valencian,

⁷This dictionary is property of the Acadèmia Valenciana de la Llengua; a portion of it has been released under a GPL license in the project page.

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<!ELEMENT orthoepikon (alphabet, annotDefSec, preannotSec, postannotSec)>
<!ELEMENT alphabet (#PCDATA)>
<!ELEMENT annotDefSec (daDefSec?, taDefSec?)>
<!ELEMENT daDefSec (daDef*)>
<!ELEMENT daDef EMPTY>
<!ATTLIST daDef n CDATA #REQUIRED>
<!ATTLIST daDef orth CDATA #REQUIRED>
<!ATTLIST daDef annot CDATA #REQUIRED>
<!ELEMENT taDefSec (taDef*)>
<!ELEMENT taDef (aDef+)>
<!ATTLIST taDef n CDATA #REQUIRED>
<!ATTLIST taDef orth CDATA #REQUIRED>
<!ELEMENT aDef EMPTY>
<!ATTLIST aDef n CDATA #REQUIRED>
<!ELEMENT preannotSec (pDefSec, dic)>
<!ELEMENT pDefSec (pDef*)>
<!ELEMENT pDef (e+)>
<!ATTLIST pDef n CDATA #REQUIRED>
<!ATTLIST pDef lemma CDATA #IMPLIED>
<!ELEMENT e (t|p)*>
<!ELEMENT t (#PCDATA|da|ta)*>
<!ELEMENT da (#PCDATA)>
<!ATTLIST da annot CDATA #REQUIRED>
<!ELEMENT ta (#PCDATA)>
<!ATTLIST ta annot CDATA #REQUIRED>
<!ELEMENT p EMPTY>
<!ATTLIST p n CDATA #REQUIRED>
<!ELEMENT dic (e+)>
<!ELEMENT postannotSec (pDefSec?, contexts*)>
<!ELEMENT contexts (annot+)>
<!ATTLIST contexts n CDATA #REQUIRED>
<!ELEMENT annot (context+|default)>
<!ATTLIST annot n CDATA #REQUIRED>
<!ELEMENT context ((t|p|ws)*, annotation, (t|p|ws)*)>
<!ELEMENT ws EMPTY>
<!ELEMENT annotation EMPTY>
<!ELEMENT default EMPTY>

```

Figure 3: DTD defining the format of the linguistic data file (see text)

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE orthoepikon SYSTEM "orthoepikon.dtd">
<orthoepikon>
  <alphabet>a-zçàèèùîâêïôû0123456789.</alphabet>
  <annotDefSec>
    <daDefSec>
      <daDef n="c_g" orth="c" annot="g"/>
      <daDef n="e_a" orth="e" annot="a"/>
      <daDef n="on_e" orth="on" annot="e"/>
      <daDef n="s_z" orth="s" annot="z"/>
    </daDefSec>
    <taDefSec>
      <taDef n="s_z" orth="s">
        <aDef n="z"/>
      </taDef>
    </taDefSec>
  </annotDefSec>
  <preannotSec>
    <pDefSec>
      <pDef n="pl1">
        <e><t></t></e>
        <e><t><ta annot="s_z">s</ta></t></e>
      </pDef>
      <pDef n="pl2">
        <e><t></t></e>
        <e><t><ta annot="s_z">s</ta></t></e>
        <e><t>e</t></e>
        <e><t>e<ta annot="s_z">s</ta></t></e>
      </pDef>
    </pDefSec>
    <dic>
      <e><t>la</t></e>
      <e><t>le</t></e>
      <e><t>le<da annot="s_z">s</da></t></e>
      <e><t>f<da annot="e_a">e</da>mme</t><p n="pl1"/></e>
      <e><t>m<da annot="on_e">on</da>sieur</t><p n="pl1"/></e>
      <e><t>se<da annot="c_g">c</da>ond</t><p n="pl2"/></e>
      <e><t>ami</t><p n="pl2"/></e>
    </dic>
  </preannotSec>
<!-- ... -->

```

Figure 4: Example of linguistic data file. Part 1 of 2: preamble and preannotation dictionary

```

<!-- ... -->
<postannotSec>
  <pDefSec>
    <pDef n="vocal">                                <!-- words starting with a vowel sound -->
      <e><t>a</t></e>
      <e><t>e</t></e>
      <e><t>i</t></e>
      <e><t>o</t></e>
      <e><t>u</t></e>
      <e><t>h</t></e>
      <e><t>y</t></e>
    </pDef>
  </pDefSec>
  <contexts n="s_z">
    <annot n="z">                                     <!-- Context in which it sounds /z/ -->
      <context><annotation/><ws/><p n="vocal"/></context>
      <!-- Followed by whitespace and vowel -->
    </annot>
  </contexts>
</postannotSec>
</orthoepikon>

```

Figure 5: Example of linguistic data file. Part 2 of 2: postannotation dictionary

to whom the annotation scheme was briefly explained, the improvement on the quality of their pronunciation when reading texts allowed was evident after introducing the annotations in the text.

guistic data file in a standard format, a system that reads in plain ISO-8859-1 text, HTML and RTF text and annotates it with marks that are easy to learn and interpret at the usual speed by almost any person reading it.

6 Concluding remarks

Orthoepikon is a set of tools to implement systems that may be used to help readers to read texts aloud when the orthography of the text is not sufficient to determine its pronunciation, when the pronunciation of the reader is influenced by another language, or when it diverges from the standard adopted for a particular communication situation. These tools, which are distributed as open-source through the Orthoepikon SourceForge site (<http://www.sourceforge.net/projects/orthoepikon>), generate, from a lin-

Acknowledgements: This project has been funded by the Acadèmia Valenciana de la Llengua (Valencian Academy of Language) and by the Vicerectorat d’Investigació, Desenvolupament i Innovació (Pro-rectorate of Research, Development and Innovation) of the Universitat d’Alacant. We thank Europa Press for kindly allowing us to use their Valencian news releases and Alicia Garrido-Alenda for her help when adapting code from interNOSTRUM.

References

- [Canals-Marote et al., 2001a] Canals-Marote, R., Esteve-Guillen, A., Garrido-Alenda, A., Guardiola-Savall, M., Iturraspe-Bellver, A., Montserrat-Buendia, S., Ortiz-Rojas, S., Pastor-Pina, H., Perez-Antón, P., and Forcada, M. (2001a). El sistema de traducción automática castellano-catalán interNOSTRUM. *Procesamiento del Lenguaje Natural*, 27:151–156. XVII Congreso de la Sociedad Española de Procesamiento del Lenguaje Natural, Jaén, Spain, 12-14.09.2001.
- [Canals-Marote et al., 2001b] Canals-Marote, R., Esteve-Guillén, A., Garrido-Alenda, A., Guardiola-Savall, M., Iturraspe-Bellver, A., Monserrat-Buendia, S., Ortiz-Rojas, S., Pastor-Pina, H., Perez-Antón, P., and Forcada, M. (2001b). El sistema de traducción automática castellano-catalán interNOSTRUM. *Procesamiento del Lenguaje Natural*, 27:151–156. XVII Congreso de la Sociedad Española de Procesamiento del Lenguaje Natural, Jaén, Spain, 12-14.09.2001.
- [Corbí-Bellot et al., 2005] Corbí-Bellot, A. M., Forcada, M. L., Ortiz-Rojas, S., Pérez-Ortiz, J. A., Ramírez-Sánchez, G., Sánchez-Martínez, F., Alegria, I., Mayor, A., and Sarasola, K. (2005). An open-source shallow-transfer machine translation engine for the romance languages of Spain. In *Proceedings of the Tenth Conference of the European Association for Machine Translation*, pages 79–86.
- [Lesk, 1975] Lesk, M. (1975). Lex — a lexical analyzer generator. Technical Report Technical Report 39, AT&T Bell Laboratories, Murray Hill, N.J.
- [Roche and Schabes, 1997] Roche, E. and Schabes, Y. (1997). Introduction. In Roche, E. and Schabes, Y., editors, *Finite-State Language Processing*, pages 1–65. MIT Press, Cambridge, Mass.