

Latin Text to Speech

Jan-Willem van Leussen Maarten Tromp

July 26, 2007

1 Introduction

In this paper we will describe our work on the creation of a Text-To-Speech (TTS) system for classical Latin. This system is suitable for any use, be it for accessibility or educational purposes, however we specifically conformed to the orthography used in the Amsterdam University Press online Latin-Dutch dictionary [2], such that it can be used as an extension of the dictionary in future.

Much is known about classical Latin pronunciation, however several details are still uncertain and possibly will always be. In our implementation we had to make a lot of decisions on how to deal with these uncertainties. Some gaps in the knowledge of classical pronunciation we filled by looking at modern Italian. For instance, we decided on vowel length and use of rhotic consonants, /r/ and /r̄/, from what is used in Italian. Of course, this could very well differ from the way Latin was spoken in classical times, however we saw it a best guess. Furthermore, an Italian sounding pronunciation probably conforms best to most people's expectations, which are likely to be influenced by the italianate Ecclesiastical Latin (the pronunciation of Latin that most often used nowadays).

We implemented our TTS system in *eSpeak* [3], an open source speech synthesizer that allows for easy creation and customization of its language modules. Most of the work in creating a language module for eSpeak does not require any programming. Instead, the specific features of a language are captured in easy to understand text files. This facilitates fast development, since no time is needed to familiarize with the source code and technical details of the synthesis process. Also this makes the language data very accessible to people unfamiliar with eSpeak or with the computer programming. This comes at the expense of flexibility though. Therefore many languages also require some programmed characteristics besides those captured in the modules. We have tried to capture as much of the data about the Latin language in the modules itself, making it easily accessible to others. This required some new syntactical constructions in the module specification, with which Jonathan Duddington, the eSpeak project administrator, helped us.

2 Features of Latin

Latin belongs to the Italic branch of the Indo-European language family. Although many people today are able to read and even converse in Latin, it is often called a ‘dead language’ in the sense that there are no longer any native speakers. This poses an obvious problem for the construction of a TTS system: how do we define a good, accurate pronunciation for a language that is not spoken natively and from which there exists no recorded speech? Fortunately there have been efforts to reconstruct the pronunciation of classical Latin as it was spoken during the time of the Roman Republic and Empire. Through careful study of the ancient texts and historical comparison with Latin’s extant descendants, the Romance language group, linguists have been able to come up with a phonological description that is thought to be fairly accurate. For our system we follow the reconstructed pronunciation of Classical Latin as described in Allen [1] as much as possible. It is fairly easy to adapt the current language modules to other varieties of Latin pronunciation, such as for instance the more modern Ecclesiastical pronunciation, which somewhat resembles Italian. The following section gives a short description of the orthographic and phonological properties of Latin that were to be implemented in our TTS system.

2.1 Orthography

To represent the Latin phonemes, the Roman alphabet (shown in Table 1) was adapted from the Etruscan, which was in turn adapted from the Greek alphabet. Unlike modern Latin texts, only capital letters were used, without any spaces or interpunction. The letters Y and Z were only used in Greek loan words. The Latin orthography that was created resembles classical Latin pronunciation very closely, mainly because the spelling was not inherited from any other language. Most letters correspond directly to a phoneme or one of a few phonemes, depending on the letter’s context. This is not the case with the vowel letters A, E, I, O and V, which all have a long and a short version, which differs both in length and quality (with the exception of the A, which only differs in length). This distinction is not captured in the standard Latin orthography. In some cases this leads to ambiguities, such as the word MALVM, which can mean ‘evil’, pronounced as [malʊm], or ‘apple’, pronounced as [ma:lʊm]. There have been various attempts to deal with this inadequacy, the oldest of which is doubling of the vowel. Later, an extra tall version of the I, called ‘I longa’ indicated the long version of the I and a diacritic called ‘*apex*’ (similar to the acute accent) marked long vowels [4]. However in classical times it was not common practice to distinguish short and long vowels in writing. Nowadays, in dictionaries and academic work, the macron diacritic is normally used to mark long vowels (ā, ē, ī, ō, ū). Sometimes short vowels are marked with the breve. For a proper pronunciation, the long vowels in the input text of our system have to be marked with macrons.

The Roman alphabet did not have the letters U and J. Instead, V was used as both a vowel (/u:/, /ʊ/) and semivowel (/w/). Similarly, the letter I was

Table 1: Latin alphabet

A	E	I	O	U
B	D	G	P	T
C	K	Q	M	N
F	S	H	L	R
X	(Y)	(Z)		

used as vowel (/i:/, /ɪ/), and as semivowel (/j/). We have adopted the modern convention, that is also used in the Amsterdam University Press Latin/Dutch dictionary, which distinguishes V/v and U/u consonantal and vowel use respectively, but used I/i for both consonantal and vowel use of I.

Specific to the Latin/Dutch dictionary is the use of tie bars in several Greek loan words (e.g. Aegēus) to indicate that the cluster of graphemes below it is monosyllabic. The on-line version of the dictionary encodes the tie bar with an acute accent (e.g. Aegéus). Also specific to Greek loan words is the ‘ȳ’ (y-macron). Since it is not present in older Unicode specifications, this character is replaced by a small image in the on-line dictionary. Our implementation cannot handle the images, and assumes a doubled y is used instead.

2.2 Phonology

The basic phonological inventory of Latin roughly contains 19 consonants, 10 vowels and 5 diphthongs. There are also a number of Greek phonemes not native to Latin that used to be utilized by some speakers nevertheless [1]. Specifically, these were three aspirated plosives /t^h/, /k^h/ and /p^h/, corresponding to the Greek letters Theta (Θθ), Chi (Χχ), and Phi (Φφ) respectively, and the close front rounded vowel /y/, written as Y. Other speakers used the native consonants /t/, /k/ and /f/ and the vowel /ɪ/ instead of the Greek phonemes. We, however, have opted to simulate the first type of speaker for our speech synthesis system.

Different sources report different sets of diphthongs of classical Latin. The letter combinations ‘ae’, ‘au’, ‘eu’, and ‘oe’ (αι αυ ευ οι) are undisputed. Some sources [1][7] also mention ‘ei’ and ‘ui’ though. If the ‘I’ is used as a consonant, the pronunciation differs only very subtly from the pronunciation as a diphthong. Compare /ɛi/ and /ɛj/. In the cases where in practice the letter combinations ‘ei’ and ‘ui’ could be diphthongs¹, the ‘I’ would be used consonantally otherwise, because is in between two vowels. Therefore, and because Harm Pinkster advised against the use of the ‘ei’ and ‘ui’ diphthongs, we did

¹this is when the letter combinations do not arise from the adjacent parts of a compound word

not implement them as such.

According to Allen, the Latin rhotic is either an alveolar flap ($/r/$) or trill ($/r̄/$). Both phonemes are produced in the same manner, except that the flap is created by one muscle contraction of the tongue, whereas the trill is created by vibrating the tongue. To decide on which phoneme to use, we looked at the Spanish and Italian use of rhotic consonants. These languages use both flap and trill, which is what we used for Latin too. The letter ‘r’ between two vowels is pronounced as alveolar flap. Other cases use the alveolar trill. Geminated ‘rr’ is pronounced as a long trill, implemented by a flap followed by a trill.

The ‘m’ at the end of a word is likely to have been pronounced weakly. This could be by devoicing it or by nasalization and lengthening of the preceding vowel. However, most references do not treat the final ‘m’ specially. For the sake of simplicity and because it is uncertain we do not treat the final ‘m’ differently from other uses of ‘m’ either.

2.2.1 Accent

Allen [1] states about word accent in Classical Latin that “there is some controversy about the nature of the historical accent, namely whether it was one of stress (as in prehistoric Latin or modern English), or of musical pitch (as in classical Greek)”, but he concludes that a stress accent is the most likely, so this is what we implemented. On the other hand the position of the accent in Classical Latin is virtually undisputed. The rules for Latin stress placement are quite simple: The accent falls on the penultimate syllable if this is a heavy syllable, and on the antepenultimate if the penultimate is a light syllable. A heavy syllable is defined as either being closed, i.e. ending in a consonant, or containing a long vowel or diphthong. For example:

virtūte	[vir.ˈtuːte]	long vowel → heavy penultimate syllable
subitō	[ˈsu.bi.tɔː]	short vowel, no coda → light penultimate syllable
fallendi	[fal.ˈlɛn.di]	short vowel, but non-empty coda /n/ → heavy penultimate syllable

Clearly, determining the word stress requires knowledge of syllabification. According to Covington [7], in classical Latin, combinations of consonants are ‘broken up’ between syllables, except for the combinations ‘ng’, ‘qu’, ‘pr’, ‘tr’, ‘cr’, ‘chr’, ‘br’, ‘dr’, ‘gr’, ‘pl’, ‘cl’, ‘bl’, ‘gl’, ‘ph’, ‘th’, and ‘ch’. Note that most of these combinations are plosives followed by liquids. The combinations ‘ph’, ‘th’, and ‘ch’ are included because they are pronounced as one phoneme. The letter ‘x’ on the contrary, is pronounced as two phonemes (i.e. $/ks/$), which are broken up. When followed by a vowel, a single consonant or a consonant cluster that is not broken up forms the beginning of a new syllable, rather than the end of the previous syllable. Here are a few examples:

annus	[an.nʊs]
actor	[ak.tɔr]
axis	[ak.sɪs]
ācris	[a:.krɪs]

3 eSpeak speech synthesis

eSpeak is a free open source software speech synthesizer, which allows users to add and customize its language modules. Its voice is normally generated by artificial synthesis but it can be connected to MBROLA for smoother sounding diphone synthesis. Although still limited at the time of writing, it offers support for Speech Synthesis Markup Language (SSML).

In eSpeak much of the language data is stored in modules, which are easily created and customized. The modules are created by compiling easy to understand text files. The benefit of this is that it does not require detailed knowledge of the speech synthesis process, or understanding of the program source code to read or modify data in the modules. Hence, it allows for easy and fast development.

4 Implementation

The following section presents an overview of the implementation in eSpeak of the characteristics of Latin, described in section 2. For a more detailed look, we recommend both reading the eSpeak documentation and examining the module source files that we have created for Latin.

4.1 Voice file

The voice file defines a number of basic settings for the language and voice in question, such as gender of the speaker, the character set used for the language, and default stress placement. The voice file for Latin sets the voice gender to male and defines the ISO-8859-4 character set (which includes vowels with macrons) as the default alternative to UNICODE. Also it sets the default stress position to the penultimate syllable, disables automatic secondary stress, and makes monosyllables unstressed.

4.2 Phonemes

To create the set of phonemes defined in `ph_latin`, we initially copied the phonemes we needed from other languages. Most phonemes are inherited from the master phoneme table (which is optimized for English), but can be overridden, if necessary. The phoneme set from German seemed a good starting point for us, since it contains most phonemes of Latin; We only had to copy the /*ɛʊ*/ diphthong from Portuguese. This provided us with a good base for starting with

the grapheme-to-phoneme conversion rules. Later we fine tuned the phoneme definitions to get a better sound. We prolonged the vowel and diphthong lengths to be similar to the vowel lengths of Italian. This we did because we felt that the length of the German vowels did not sound ‘Romance-like’. We also copied the rhotic phonemes /r/ and /r/ from Italian to replace the German rhotic. Furthermore, we set the vowel formant frequencies to more or less average values for the phonemes, as opposed to the German versions. Also we defined an unaspirated /p/ to override the default aspirated version (as in English) and we changed the /l/ (see below). The labialized consonants /g^w/ and /k^w/ are implemented as the consonant followed by a /w/ phoneme. Similarly, in order to be able to produce the aspirated phonemes /k^h/, /t^h/, and /p^h/ we defined a very short /h/ phoneme, that is meant to succeed another phoneme, to create the aspiration effect.

eSpeak stores the information about the phonemes in a ‘phoneme table’ text file. All phonemes are represented in ASCII characters using the Kirshenbaum scheme [8]. The phoneme tables allow the user to define the phonemes of a particular language by type (vowel, liquid etc), length in milliseconds, and formant frequencies. A prerecorded WAV file may alternatively be used for some phonemes that are difficult to synthesize. Allophonic pronunciations can also be defined. We used this to produce a ‘dark l’ (/ɫ/) at the end of a word. Figure 1 below is the definition of the /l/ phoneme, including the variation /ɫ/ (defined here as l/2) that is heard when the /l/ is in word-final position or preceding a consonant.

```

phoneme l
  liquid
  length 100
  lengthmod 7
  beforenotvowel l/2
  formants l/l
  after _ l/_l
  after t l/tl
  after l/ l/l_long
endphoneme

```

Figure 1: Example of a phoneme definition in eSpeak

In this manner we have been able to define all Latin phonemes as listed in Appendix A. However, most of the phonemes will most likely still need some tweaking to approach a better reconstruction of spoken Latin. This may require a careful evaluation of the current state of our system by an expert on Latin phonology. The aspirated phonemes specifically can be synthesized more elegantly.

4.3 Grapheme-to-phoneme conversion

After defining the phonemes, the next task was to define their correspondence with the characters of written Latin. As mentioned in the section on orthography, there are some different approaches to writing Latin, e.g. the use of macrons, the use of ‘u’ or ‘v’ for non-consonantal V, etcetera. We conformed to the orthographic standards of the Latin-Dutch dictionary [2], as it will be the primary input for our TTS system.

Grapheme-to-phoneme conversion in eSpeak is done according to a set of context-sensitive rewriting rules accompanied by a dictionary of exceptions to these rules. For Latin, these are stored in the ‘la_rules’ and ‘la_list’ files respectively. The latter of these two files also contains lists of abbreviations, numerals, and function words that need to be treated differently with respect to sentence prosody and rhythm. The rewriting rules (stored in the ‘la_rules’ file) allow the user to define pronunciations for a single character or a group of characters, given a certain context. Figure 2 shows, as an example, the rules for the Latin grapheme G.

```
.group g
      g           g
      gg          g:
      g (n        N
      gu (A       gw //g<w>
```

Figure 2: Example of grapheme-to-phoneme rules in eSpeak

We have been able to capture all pronunciation rules described by Allen [1], including word stress, in the rules file. We included an extra pronunciation rule that is not described in the literature, but does improve the sound significantly. Namely, we inserted a glottal stop between two consecutive ‘i’s. We noticed that /ii/, as pronounced by eSpeak sounds very unnatural. It seemed reasonable to us that it used to be pronounced with a glottal stop in between, /iʔi/, which indeed sounds much better.

The list file does not need to define any irregular pronunciations, since Latin is an extremely regular language. It does define the pronunciation of letter names and numerals though.

Normally, the grapheme corresponds with the phoneme /g/ (unvoiced velar plosive). A doubled g (or other consonant), however, stands for a geminated (lengthened) g. A G followed by the grapheme N represents the phoneme /ŋ/ (velar nasal), thus magnus ‘great’ is pronounced [maɲnʊs] and not [magnʊs]. The capital A on the last line is a special symbol matches any vowel. Hence, the combination GU, when followed by a vowel, represents a labialized velar plosive, and is pronounced /gw/ rather than /gu/. When the grapheme-to-phoneme conversion is confronted with multiple possible rules for a word, more specific rules are preferred by eSpeak. The user can also manually determine a preference for certain rules above others.

Apart from the capital A, there are more such special symbols which can be used in the context part of rules. The ‘@’ symbol for instance states that a vowel follows somewhere in the rest of the word if it appears on the right side of the rule. Despite the existing range of special symbols, the scheme was not powerful enough to express the rules that determine the proper word stress. Jonathan Duddington, the administrator of the eSpeak project, helped us extending the scheme with two new special symbols, which make it a lot more powerful. First, there is the X symbol, which matches if there are no more vowels following. Second, there is the option to match sets of user-defined letter sequences. This made it possible for us to implement our language without having to add Latin specific subroutines to the program itself. The new capabilities are likely to be very useful for other languages too.

Syllabification and word stress assignment are the tasks for which we needed the new special symbols. These can now be performed by our rules entirely. Writing down the rules without the new capabilities is not very tractable due to the combinatorial explosion of cases to cope with.

By default, word stress is assigned to the penultimate syllable of all words (as defined in the voice file). In the case of weak penultimate syllables, stress is placed on the antepenultimate syllable. Figure 3 is an example of some of the rules that decide whether a syllable is weak.

```
.group I
      I           I
// Weak penultimate syllable: put stress on antepenultimate
      I (L01X    =I
      I (CL01X   =I
      I (L02L01X =I
      I (guL01X  =I
```

Figure 3: Determination of word stress through context, for syllables containing a short I.

In the left column, the symbols L01 and L02 stand for ‘any vowel/diphthong’ and ‘any plosive+liquid combination’, respectively; X means ‘no more vowels until the word boundary’, and C means ‘any consonant’. In the right column, an equation symbol (=) preceding a phoneme means ‘accent on preceding syllable’. The bottom four rules in Figure 3 all describe situations in which a penultimate syllable containing an I has no coda, and should therefore not be stressed. We have been able to define the stress rules for all possible phoneme combinations of Latin in this way.

4.4 Sentence stress

With the basics of grapheme-to-phoneme conversion and word stress covered, another step toward a natural-sounding synthesis is improving the sentence prosody. Unfortunately this is a notoriously difficult problem, as choosing the

right word to place the focus on is largely determined by semantic cues. The relatively unfixed word order of Classical Latin also means that some morphosyntactic analysis would be necessary to correctly determine which words in a sentence form constituents sharing prosodic features. However such analysis requires detailed knowledge of Latin syntax and would be beyond the scope of this project.

Nevertheless we have been able to somewhat ameliorate sentence accent by manually constructing a list of words in the ‘list’ dictionary file which are likely to be unstressed, such as certain pronouns, prepositions, and forms of the copula *esse*. Likewise we have utilized the possibility to add a short pause before certain words, e.g. conjunctions like *et* ‘and’, *ergo* ‘thus’ and so on. Unfortunately even the approach listing all words which are likely to be unstressed is not always feasible. For example ‘*cuius*’ (‘whose’) may, like its English counterpart, both be used as an interrogative and a relative pronoun; the first should probably often be stressed, the latter should not. We have chosen *not* to mark words such as these as unstressed.

4.5 MBROLA voice

As mentioned earlier, eSpeak does not only allow customization of voices using its native artificial synthesis, but can also be linked to an MBROLA [6] diphone voice file. Diphone synthesis is generally more agreeable and natural-sounding than full artificial synthesis. The creation of an MBROLA diphone voice is a long process, as all possible two-phoneme combinations of a language have to be recorded by a speaker. Fortunately a voice file created by Olivier Bianchi, who is using it as a basis for another Latin TTS system (cf [5]), is available for free from the MBROLA project web site.

eSpeak can be used as a front-end, first converting text to a phonetic representation using the normal ‘phonemes’, ‘rules’ and ‘dictionary’ files, and then translating this representation to the MBROLA phonemes. This translation is made through a simple file which eSpeak uses to link each phoneme defined for a language to an MBROLA phoneme. Although it is not possible to customize the MBROLA voice much as the files are pre-recorded, and concatenative synthesis sometimes results in small ticking sounds between certain phoneme combinations, we found that overall using the MBROLA voice greatly improved the quality of our TTS module, compared to the more ‘robotic’ sounding artificial synthesis.

5 Future work and conclusion

In this report we have described the process of creating a basic Latin TTS module for eSpeak. Although some changes have been made to the eSpeak source to meet the needs of our TTS module, most of the language rules are actually defined in a number of easily customizable text files. While we believe the system to be already of acceptable quality, with pronunciation and stress

mostly correctly implemented, the system could still be improved considerably in a number of ways:

Phoneme quality of the eSpeak voice may be improved by tweaking duration, formants, and allophonic variants. However, this will not affect the MBROLA voice, which at the moment sounds considerably better than the eSpeak voice and is probably preferred for the online dictionary and other applications.

It would also be possible to add other pronunciation standards, for example an Ecclesiastical Latin voice or a ‘Dutch/Germanic Latin’ voice. This would mostly be a simple matter of changing the phoneme definitions.

Sentence prosody may be improved through morphosyntactic analysis and refining of the ‘list’ dictionary files. Support may be added for the correct pronunciation of Roman numerals (both ordinal and cardinal), abbreviations and the like. As these often decline according to syntactic function, here again some morphosyntactic analysis of text would be needed to approach a correct pronunciation.

At the moment, correct pronunciation of Latin texts with our module requires that they be marked for vowel length with macrons; unfortunately this is far from common practice. Since vowel length in many cases is largely systematic (e.g. for noun and verb endings) it would be a feasible and worthwhile addition to make automatic annotation software, perhaps by using a learning algorithm to automatically deduce rules for macron placement.

Acknowledgements

We thank Jonathan Duddington, the author of eSpeak, for his quick advice and considerable help, which included updating the program to specifically meet our needs. Also we thank Olivier Bianchi, whose Latin Mbrola voice we used for this project. Finally we thank Harm Pinkster, on whose incentive this project was started, for taking time to answer some of our questions about the Latin language.

References

- [1] Allen, W.: *Vox Latina. The pronunciation of Classical Latin*. Cambridge University Press (1989)
- [2] Pinkster, H.: *Woordenboek Latijn/Nederlands*, Amsterdam University Press (2003), <http://www.latijnnederlands.nl/>
- [3] *eSpeak* open source software speech synthesizer, <http://espeak.sourceforge.net>
- [4] Oliver, R. P.: ‘*Apex and Sicilicus*’, *American Journal of Philology* 87 (1966), pp. 129 ff.
- [5] Bianchi, O. and Keller, E.: *Virtual Historic Reconstruction with Speech Synthesis*, in Braun A. et Masthoff H.R. (eds), *Phonetics and its Applications. Festschrift for Jens-Peter Köster on the Occasion of his 60th Birthday*, *Zeitschrift für Dialektologie und Linguistik, Beiheft 121*, Stuttgart : Steiner, 465-484, (2002)
- [6] MBROLA project homepage, tcts.fpms.ac.be/synthesis/mbrola.html
- [7] Covington, M.: *Latin pronunciation demystified*, <http://www.ai.uga.edu/mc/info.html> (2005)
- [8] Kirschbaum, E.: *Representing IPA phonetics in ASCII*, <http://www.kirshenbaum.net/IPA/ascii-ipa.pdf> (1992)

Appendix

A Phonemes of Latin

Consonants

IPA	SAMPA	Kirschenb.	grapheme	description
b	b	b	b	voiced bilabial plosive
p	p	p	p	voiceless bilabial plosive
m	m	m	m	Bilabial nasal
f	f	f	f	voiceless labiodental fricative
d	d	d	d	voiced alveolar plosive
t	t	t	t	voiceless alveolar plosive
z	z	z	z	voiced alveolar fricative
s	s	s	s	voiceless alveolar fricative
n	n	n	n	alveolar nasal
r	4	*	r	alveolar tap
r	r	r<trl>	r	alveolar trill (rolling R)
l	l	l	l	alveolar lateral approximant
ɫ	5	l<vzd>	l ¹	velarized alveolar lateral approximant
j	j	j	i	palatal approximant
g	g	g	g	voiced velar plosive
k	k	k	c	voiceless velar plosive
ŋ	N	N	n ²	Velar nasal
k ^w	k_w	k<w>	qu	Voiceless velar plosive (labialized)
g ^w	g_w	g<w>	gu	Voiced velar plosive (labialized)
w	w	w	v	voiced labial-velar approximant
h	h	h	h	voiceless glottal fricative
k ^h	k_h	k<h>	ch	aspirated voiceless velar plosive
t ^h	t_h	t<h>	th	aspirated voiceless alveolar plosive
p ^h	p_h	p<h>	ph	aspirated voiceless bilabial plosive

¹as a final 'l', i.e. at the end of a word

²before the graphemes 'k', 'q', or 'c'

Vowels

IPA	SAMPA	Kirschenb.	grapheme	description
a	a	a	a	open front unrounded
a:	a:	a:	ā	open back unrounded
ɛ	E	E	e	open-mid front unrounded
e:	e:	e:	ē	close-mid front unrounded
ɪ	I	I	i	near-close near-front unrounded
i:	i:	i:	ī	close front unrounded
ɔ	O	O	o	open-mid back rounded
o:	o:	o:	ō	close-mid back rounded
ʊ	U	U	u	near-close near-back
u:	u:	u:	ū	close back rounded
y	y	y	y	close front rounded vowel ¹
y:	y:	y:	ȳ	close front rounded vowel ¹

¹Appears in Greek loanwords only

Diphthongs

IPA	SAMPA	Kirschenb.	grapheme
aɪ	aI	aI	ae
aʊ	aU	aU	au
ɛʊ	EU	EU	eu
ɔɪ	OI	OI	oe