Toward Language-independent Text-to-speech Synthesis

MARIO MALCANGI, PHILIP GREW
DICo – Dipartimento di Informatica e Comunicazione
Università degli Studi di Milano
Via Comelico 39 – 20135 Milano
ITALY
malcangi@dico.unimi.it grew@dico.unimi.it http://dsprts.dico.unimi.it

Abstract: - Text-to-speech (TTS) synthesis is becoming a fundamental part of any embedded system that has to interact with humans. Language-independence in speech synthesis is a primary requirement for systems that are not practical to update, as is the case for most embedded systems. Because current text-to-speech synthesis usually refers to a single language and to a single speaker (or at most a limited set of voices), a framework for language-independent, text-to-speech synthesis is proposed to overcome these limitations in implementing speech synthesis on embedded systems. The proposed text-to-speech synthesis framework was designed to embed phonetic and prosodic information in a set of rules. To complete this language-independent speech-synthesis solution, a universal set of phones has been defined so that the appropriate speech sounds for every language are available at run time. Synthesis of more than one language can easily be carried out by switching from one rule set to another while keeping a common phone-data set. Using a vocal-track-based speech synthesizer, the system does not depend on phone sets recorded from an actual specific human voice, so voice types can be chosen at run time.

Key-Words: - Text-to-speech, multi-language speech synthesis, rule-based speech synthesis

1 Introduction

Concatenative text-to-speech (TTS) synthesis has gained widespread market acceptance in recent decades as a practical solution for embedding unlimited speech-production capacity in a system [1]. This solution has the advantages of unlimited vocabulary and not requiring new strings to be uttered by a human speaker. However, its major limitations include not only the use of a predefined speaker's voice but also the fact that a system is language-specific, i.e. any particular implementation of such a solution is dedicated to a single language.

Current concatenative TTS synthesis is based on patterns for storing speech (diphones, demisyllables, etc.) and on sets of rules that describe how the patterns are to be concatenated to produce a version of the natural utterance corresponding to a specific alphabetical text[2]. Very high-quality speech can be obtained from this approach, though at the expense of system flexibility. The number of diphones in a given language ranges into the thousands. Each human voice to be sampled must be represented by several hours of recorded speech that has been preprocessed and stored in a database.

The main limitation of concatenative synthesis lies in the way the speech segments to be concatenated are obtained. Diphones or demisyllables are strictly dependent on language and highly redundant. Diphone or demisyllable sets require huge amounts of memory to completely cover all the phonetic variability of the target language. If the synthesizer is to speak a different language, a new diphone or demisyllable set needs to be derived from a language-specific utterance database. Therefore, if the system application has to run a bilingual application, system memory requirements double at the very least.

Another limitation is due to how diphones or demisyllables are produced. These are derived from the utterance of a specific speaker by means of a speech- editing and analysis process. If a different voice has to be added to an application, twice as much memory must be made available. Diphone systems typically limit the number of phone variants available for the obvious reason that each variant might potentially need to be present in storage in a number of occurrences equal to that phone's possible co-phones. Another well-known drawback of diphone systems is the obstinacy of diphone-boundary artifacts, which has driven some researchers to reconsider the viability of unit-selection synthesis as an alternative [3].

Like the issue of inter-unit boundary artifacts, the question of how best to modulate synthesized speech output so as to incorporate prosodic features also suffers from limitations inherent in the concatenative process. Demisyllable synthesis

holds promising potential flexibility, with the option of including non-segmental features at a lower (earlier) level of the synthesis process. However, like diphone synthesis, it has been around for a long time yet still provides high-quality results in both naturalness and intelligibility only at great expense of resources and specific tuning. For the time being, this appears to remain the case despite the attractive, underlying simplicity of the methods demisyllable synthesis involves [4].

Next-generation, embedded-system applications will need multi-lingual, multi-speaker options, but will offer the developer meager system resources, because the systems that host such applications will increasingly tend to be embedded. Clearly, this kind of environment is not ready to host today's state-of-the-art TTS engines, although there is some ongoing research that aims to turn this challenge around by insisting that, vice-versa, the state of the current TTS art needs to be readied for the next-generation environment.

As a result, alternative approaches [5, 6, 7, 8, 9, 10, 11] to TTS synthesis strive to overcome the limitations of current TTS synthesis by enabling the developer to meet next-generation product requirements. We are proposing a TTS framework based on phonemes, allophones and pseudo-phonemes, essentially phones, collected from different languages. Upon this phone set is erected a set of language-dependent, text-to-speech rules. Our hybrid framework then uses a formant-based speech synthesizer to generate speech elements to be assembled [12], so that speaker dependency can be avoided. This leads to a deeply-embeddable multi-language, speaker-independent, TTS-synthesis model.

However, a sliding-window approach, applied from the moment that grapheme-to-phoneme conversion is first undertaken, not only replaces some of the a posteriori smoothing typically required to overcome segmentation artifacts but also affords a natural potential input venue for integrating non-segmental information. Intonation instructions, i.e. specification of what direction pitch is changing in and how fast, need not be applied to the same size sample as the segment fed to the transcription engine or the phone-sequence-to-sound-sequence synthesizer module.

Encoding emotion from the ground up has long been expected to be one of the upcoming challenges for TTS synthesis. To the extent that emotional signals are predictable from written text alone, which is perhaps greater than one might expect [13], it is largely a matter of pauses, changes in pitch, and variation in stress. Recent strides in emotion recognition have relied on feature-extraction methods that have been shown (in [9] and elsewhere) to be fully compatible with our TTS framework [14].

In addition to the aim of obtaining naturalness in TTS systems as envisioned in [13], one significant objective of encoding emotion into synthesis rules is to generate patterns to be matched during queries of recorded voice data [15]. To this end, the emotion-feature definitions available in [14] would appear ready-made for low-level programming (i.e. direct device control) into the synthesis model proposed here.

2 Transcription Notation

The first operation performed on text that is input into this system is grapheme-to-phoneme conversion, i.e. transcription. Although white space, punctuation, paragraph breaks, and the like do not produce any actual phonemic representation, this non-grapheme information is not totally ignored, merely set aside for further possible processing when suprasegmental features can be integrated into the synthesis module's actual articulation. For example, the rising intonation represented by a question mark in the original orthographic notation used as input will not show up in an initial phonemic transcription or a subsequent phonetic transcription but nevertheless represents information to be used during articulation.

The bare-bones phonetic transcription that is to be the output of this first operation of the rules on the input will thus become input for subsequent processing. Obviously, therefore, such a transcription will not look like the phonetic transcription one might find in a dictionary. There are two reasons for this, one notational and one notional.

2.1 The notational issue

The notational reason lies in the need for transcription-engine output to be machine readable. Obviously, save for human-readability, there would be no advantage in attempting to transcribe the input's orthographic notation into some sort of phonetic alphabet such as the international phonetic alphabet (IPA) [16]. Furthermore, such a transcription would be font-dependent, making it useless for the purposes described here.

Of course, there have been adaptations of IPA into ASCII, although the human-readability of such systems is, at times, compromised by conflicting

Concise description of sound	*Corresponding IPA symbol*	*X-Sampa-like encoding base*	*Language of sample*	*Sample word*
schwa	ə	@	English	(a)bout
near-open front unrounded vowel	æ	{	English	m(a)p
long near-open front unrounded vowel	æː	{:	English	m(a)d
close-mid front unrounded vowel	e	e	Italian	sc(e)lte
open-mid front unrounded vowel	ɛ	E	Italian	b(e)lla
mid front unrounded vowel	ɛ̝	E_r	Italian	b(e)nché
close front unrounded vowel	i	i	Italian	v(i)sti
open front unrounded vowel	a	a	English	(o)dd
close back rounded vowel	u	u	Italian	p(u)nto
close-mid back rounded vowel	o	o	Italian	p(o)ngo
open-mid back rounded vowel	ɔ	O	Italian	(o)rto
mid back rounded vowel	ɔ̝ *	O_r	Italian	c(o)priletto
near-close near-front unrounded vowel	ɪ	I	English	h(i)t
open back unrounded vowel	ɑ	A	English	f(a)ther
voiceless bilabial plosive	p	p	Italian	(p)ongo
geminate stop	pp	pp	Italian	ca(pp)otto
geminate stop	pːp	p:p	Italian	stra(pp)o
aspirated	^{h}	_h	English	p()ail
voiceless glottal fricative	h	h	English	(h)eart
hold fricative closure		H_f	English	chic(k)en
hold nasal		H_n	English	uh-(h)uh
hold vocal		H_v	English	ba(h)
hold vocal closure		H_c	English	d()oes
glottal stop	ʔ	?	English	Clin(t)on
alveolar flap	ɾ	4	Italian	ie(r)i
alveolar trill	r	r	Italian	(r)aro
geminate sonorant	rːɾ	r:4	Italian	ca(rr)o
geminate sonorant	ɾr	4r	Italian	co(rr)esse
geminate sonorant	ɾɾ	44	Italian	dà (r)agione
alveolar approximant	ɹ	r\	English	(r)ed
voiced uvular fricative	ʁ	R	French	(r)oi
uvular trill	ʀ	R\	German	D(r)ang
palatal nasal	ɲ	J	Spanish	a(ñ)o
labiodental nasal	ɱ	F	Italian	go(n)fio
open-mid front rounded vowel	œ	9	French	n(eu)f
falling back rounded diphthong	oʊ	oU	English	b(oa)t
open-mid back unrounded vowel	ʌ	V	English	c(u)t
near-close near-back rounded vowel	ʊ	U	English	f(oo)t
open back rounded vowel	ɒ	Q	Br. Eng.	cl(o)ck
close front rounded vowel	y	y	French	t(u)

Table 1 – Partial phone set for initial, human-readable, grapheme-to-phoneme transcription notation. Parentheses enclose the letter that 'makes' the phone in question. Shaded phones represent the nine fundamental vowel qualities used for Italian TTS (see section 4.1). The symbol marked * is an extension of IPA used by [22].

standards. For example, Pronlex [17] used the symbol @ to represent /æ/ while others, like Kirshenbaum [18], used it for /ə/, i.e. schwa. Pronlex and another commonly found ASCIIfication of the IPA, one-letter Arpabet [19], use x to represent schwa. The single-character transcriptions were designed to allow linguists to communicate in email and newsgroups, rather than for machine processing.

To the extent that our TTS rules need to be human-readable (say, so that a native speaker of the language being synthesized can more easily tune the system), we have based the transcription standard on X-SAMPA [20], though what ultimately must become the final input for the synthesis module is a complex set of numbers representing the synthetic utterance. The phonetic information included those numeric strings is combined with other voice data to control a series of parameters. It is, of course, not human-readable at all.

The history of TTS research is littered with competing, fully machine-readable transcription systems, so describing their differences would go well beyond the scope of this paper. Suffice it to point out that many of these systems, like two-letter Arpabet, use two-character, case-insensitive ASCII text, with delimiters between phones, to produce a transcription that is less human-readable but more practical for machine processing. The two-character phone sets (though not Arpabet) often include numeric characters, as well (which complicates human reading of transcription output when the phones are marked numerically for length, as in our case). Although what counts is the ability of any system to be easily converted into another, the framework described here has been developed using transcription rules based on a notation that delimits phones with slashes and prefixes each phone with a number indicating its duration. The system we have applied is based on the case-sensitive X-SAMPA phoneme set, with the addition of a few extended symbols, including some for non-phonemic features such as holding an articulatory closure.. X-SAMPA is also used for the phonetic notations enclosed in a single pair of slashes hereinafter.

2.1 The notional issue

The notional reason why the transcription produced by an initial application of rules need not resemble human-dictionary phonetics is that the notional value of the information represented is not tied to traditional segment sets. The conceptual distinction between a feature distinctive to the phoneme and one that is allophonic is not germane to the processing of these rules. The input text string in orthographic notation must be turned into the correct acoustic product but its sounds do not necessarily have to be analyzed in traditional linguistic terms.

Segmental and suprasegmental features will ultimately be blended in the final instruction set sent to the synthesizer module. Often a phone will not really reflect its belonging to (part of) a given phoneme as much as being one member of the combination of sounds that belong to an overall utterance. The audio output will, ideally, be a flow of blended components. Segments are inevitable facts of life only at the time of input consisting of alphabetical segments. From the start, rules are going to take more than one segment into account. To a certain extent, then, the traditional notion of the phoneme can be ignored in the process of overcoming the limits of orthographic notation. The problem of spelling irregularity pales by comparison to the risks inherent in basing a rule "on a phonological unit that is arguably not a natural unit" [21]. In other words, one of the great limits of TTS, aside from the instance of an actual human being reading aloud, is imposed by the alphabet itself. Tying the transcription engine to a particular notion of the phoneme is likely to reintroduce the limitations of the alphabet at precisely a stage of the process in which these need to be overcome.

Therefore, we have attempted to apply to the encoding of TTS rules a transcription notation that will, as far as possible, not be bound by traditional notions of segmental, phonemic or allophonic production of sound from its character-based representation. That having been said, of course the transcription used for encoding owes much to previous systems and methods. And, it too, takes a set of likely phonemes as its starting point. Table 1 shows some sample words the initial phonemic representations employed for the sounds contained in them, where the sound in question 'belongs' to the letter in parentheses.

3 System Framework

The system framework, as illustrated in figure 1, consists of three main components: the rule-based language model, the multi-language phonetic data, and the formant-based speech synthesizer.

The rule-based language model (see figure 1, below) is a collection of language-specific sets of rules. Each set collects all the phonetic information needed to transform a string of alphabetical text into the correct utterance for a specific language.

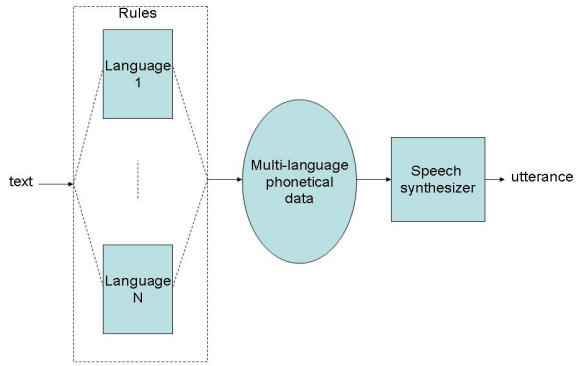

Fig. 1 – System framework for a multi-language text-to-speech synthesizer

The multi-language phonetic data is a superset of the phonetic data related to all the languages to be synthesized. It need not be limited to what are traditionally considered features of the phoneme but can include stress or intonation patterns, as well.

The formant-based speech synthesizer is an artificial model of human ability to produce speech signal from phonetic and linguistic information. It is a parameter-driven synthesizer capable of generating all the phonetic samples included in the multi-language phonetic data set.

4 Rule-based Language Model

To model a language in terms of phonetics, we use a rule-based model whose rules are regular expressions such as:

$$L(S)R = /p/$$

where

L: left context
R: right context
S: string to be matched
p: phonetic sequence

Using these expressions, it is possible to encode in a set of rules all the phonetic information implicit in an alphabetical sequence. An algorithm was developed to parse the rule set and match the most appropriate rule to a specific alphabetical string. The algorithm runs on the following classes of elements defining, in terms of regular expressions, the left and right context:

(!) | (^) | ($)
(#) | ([AEIOUY]+)
(:) | ([^AEIOUY]*)
(+) | ([EIY])
($) | ([^AEIOUY])
(.) | ([BDGJMNRVWZ])
(^) | ([NR])

!	any non-alphabetical character
#	single or multiple vowels
:	zero or more than one consonant
+	one front vowel
$	one consonant
.	one voiced consonant
^	N or R consonant

These context symbols enable a compact representation of the rules, so that a large number of alphabetical character combinations can be represented by a single rule. Ultimately, this is not truly a 'pure' rule-based system because the least generally applicable 'rules' amount to exceptions and, as such, may contain what amount to lexical items (as can be seen in the second of the following three examples). This aspect of our rule set bears nearly enough resemblance to dictionary-based transcription to be thought of as a hybridization. However, because no external lexical database need be applied, the model remains rule-based in nature.

Should future prototypes of this TTS framework take the step of spinning lookup tables of lexical items off from the primary language-specific rule set (say, for purposes of speeding up rule parsing), it is important to bear in mind that the resources required for tagging, storing, retrieving, comparing, and re-scripting an extremely huge number of lexical items still pale in comparison to the resources demanded by similar management functionality for even a very modest database of diphones.

As a matter of fact, much of what seems at first blush to be language idiosyncrasy proves, upon analysis, to be only a minor challenge for a rule-based system.

4.1 Treatment of Italian {e}

The following example shows a surprisingly simple solution to a well-known instance of a breakdown in what native speakers usually expect to be a biunique mapping of grapheme to phoneme. This series of rules, albeit not exhaustive, matches the letter {e} to its phonic realization in Italian words (and it should be borne in mind that # represents a vowel letter and

not a word boundary, while we follow convention in using braces to indicate orthographic notation):

!(E)! = /2e/| Isolated {e} is stressed /e/

!(E)$$#! = /2e/| Word-initial {e} followed by two consonants and by one or more word-final vowels is short stressed /e/

!(E)$#! = /2e:/| Word-initial {e} followed by one consonant and one or or more word-final vowels is long stressed /e/

!(E)$#$ = /2E/| Word-initial {e} followed by one consonant and one or or more pre-consonantal vowels is stressed /E/

!(E)$:#! = /2E/| Word-initial {e} followed by one consonant, a non-vowel and a word-final vowel is stressed /E/

!(E)$# ! = /2E/| Word-initial {e} followed by one consonant and a word-final vowel is stressed /E/

!(E) = /1e/| Generic word-initial {e} is unstressed

(E)! = /1e/| Word-final {e} is unstressed unless accented with a diacritic

These rules solve three key issues specific to the Italian language, issues that are glaring in Italian because, exceptionally for this language, orthography gives no clue as to which of several possible pronunciations is the norm. In other words, speakers rely on tacit knowledge derived from context to chose the proper phonetic realization of a given alphabetic string (as commonly occurs in English, for example). The three issues being solved are the following:

- whether the {e} vowel is open or closed;
- whether the {e} vowel is stressed or unstressed;
- whether the {e} vowel is long or short.

Current TTS methods solve such problems by means of language-related algorithms that have to be coded each time for a specific processor and for a specific language. Paradoxically (and perhaps confirming some of what is claimed in [21]), the above example illustrates how grapheme-to-phoneme conversion would introduce and unnecessary issue.

A great deal of linguistic research has been devoted to the issue of how many vowel phonemes are actually used in modern Italian. The five vowel letters have generally been considered to correspond to seven vowel phonemes, i.e. including open or closed {e} and open or closed {o}. Our rule set does not actually attempt to convert the graphemes into phonemes. Rather, context is used to determine both length (duration) and to choose between three (!) different vowel qualities for each of the letters {e} and {o}.

The grapheme-to-phone-string transcriptions encoded in the rules above thus follows the set of vowels in the "neutral Italian" defined by [22] in attributing three different vowel heights to the letters {e} and {o}. In other words, for synthesis purposes, it does not matter just where the distinction among vowel type becomes phonemic. What counts for the ear is that the proper quality be given to the basic phone and that length be adjusted as a function of stress. Therefore, while our work says nothing about actual the number of vowel phonemes in Italian, it does require a phone set that includes nine vowels, thus supporting arguments made in [22] for raising and lowering two of the seven traditional phonemes.

4.2 The English past-tense morpheme

The example below shows how rules can be applied to match phones to word-final {ed} in English text. This group of rules, though hierarchical, is not exhaustive. For example, the entire set of words in {-ed} in which this string is part of the root and not the past morpheme (i.e. *naked*, *jagged*, *rugged*, *ragged*, *coed*, and very few others) is represented by the rules for *rugged* and *coed*. As is well known, the phonetic realization of {-ed}, when this orthographic notation does indeed correspond to the past-tense marker (as in the vast majority of cases), depends on the final sound of the base form of the verb. In practice, orthographic notation, i.e. standard text, gives us enough information to predict which phonetic realization matches the word-final text string {ed}.

Although there will always be some word-specific exceptions, which have to be mapped to longer strings ordered at the top of the hierarchy (i.e. with {rugged} and {coed}), the ordering of the rules enables a rather economical mapping of the three cases that account for the overwhelming majority of text-strings in which {ed} is followed by a non-alphabetical character, a realization as /Id/, as /d/ or as /t/. It is worth noting that because these ordered rules treat rarest cases first, the most common realization, /t/, is generated last, obviating the need for complex rules to identify the variety of

orthographic notations that represent unvoiced contoids that were word-final prior to {-ed} suffixation. In other words, clusters like {-tch} and {-sh} need not be specifically treated:

!RUGG(ED)! = /1I/1d/

!CO(ED)! = /2E/2d/

!:(ED)! = /2E/2d/ (e.g. red, bed, Ted)

#:(IED)! = /1i/1d/ (e.g. studied, muddied)

:(IED)! = /1a/1I/1d/ (e.g. lied, tried)

(TED)! = /1t/1I/1d/ (e.g. hated, lasted)

(DED)! = /1d/1I/1d/ (e.g. aided, added)

#(ED)! = /1d/ (e.g. played, hoed)

.(ED)! = /1d/ (e.g. rubbed, hugged)

(ED)! = /1t/ (e.g. hopped, baked, itched)

A similar set of rules, reflecting different but analogous phoneme classes, as outlined in [12] (page 388) will apply to the phonetic interpretation to be given to word-final {s}.

4.3 Aspiration of initial stops in English

A final example, again from the English rule set and again involving more than one acoustic realization for a given orthographic token, stands in contrast to the above example for two reasons. The first contrast relates to what might be termed "the intuitive economy of rules." The second contrast relates to the phone-phoneme distinction and shows the limitations of this distinction for synthesis purposes.

English has a well-known (at least among linguists) grammatical feature that results in word-initial voiceless stops being aspirated. The {p} in *pail* [p^h] is thus not interchangeable with the {p} in *apple* [p], despite the fact that this distinction is not phonemic in English. The former is aspirated, whereas the latter is not. However, this allophonic variation is not specific to /p/. It occurs in exactly the same context with /k/ and /t/, as well.

A rule-based synthesis system might be expected to include a processing rule to reflect the grammatical rule. This conveniently proved to be the case for the phonetic treatment of the past-tense morpheme {-ed}. But the processing rule set contains no rule that reflects the concept "word-initial voiceless stops are aspirated." Rather, this phonological rule is applied independently to the three separate phonemes. Furthermore, there is no need to apply this grammatical rule during the first grapheme-to-phoneme conversion step, while there are clear advantages to encoding aspiration at a subsequent stage when stress in encoded. This is a logical point at which to tune release features, and it is when the first Italian stop in a post-tonic geminate is lengthened.

The salient rules for encoding {p} are as follows (note that the dash in a phone slot represents a pause and bear in mind that numbers preceding the phone indicate duration):

!(P)!=/1p/1i/

(P)!=/1p/4h/4-/

(PA)STE=/1p/1eI/1j/

!(PHOTO)=/1f/1o/1w/1t/2o/2w/

!(PHYS)=/1f/2I/1z/

(PH)=/1f/

(PPH)=/1f/

(PEOP)=/1p/1i/1p/

!(POE)T=/1p/1o/1E/

(POUR)=/1p/1o/13/

(POW)=/1p/1O/1u/

(PP)=/1p/

!(PRETT)=/1p/1r\/2I/1t/

(PRO)VE=/1p/1r\/1u/

(PROO)F=/1p/1r\/1u/

(PRO)=/1p/1r\/1o/

(PSEUDO)=/1s/2u:/2d/3o/3w/

(PSYCH)=/1s/2a/2a/3j/1k/

!(PS)=/1s/

!(PT)=/1t/

CEI(PT)=/1t/

(PUT)!=/1p/1U/1t/4-/

!(P)=/1p/2H_f/

(P)=/1p/

Even a cursory glance at these rules, bearing in mind that they are ordered (and that this set is not complete, a few of the less significant having been deleted for reasons of space) shows they operate at a fine level of granularity. For example, it is easy to identify the rule that will be used solely to encode stings like {poet} and {poetry}. Since this rule is limited in its application to contexts in which some sort of white space preceded the string, we can be pretty sure the {p} in them is word-initial. It would be easy to encode it as /1p_h/ rather than /1p/ but this information is not really helpful at this stage. Better to wait for the aspiration until prosodic features and articulation instructions give rise to a natural point at which to integrate a less segmental aspiration (with its own duration that is not necessarily co-terminant with that of the stop) . Therefore these rules assign the {p} in word-initial {poet-} the same transcription /p/ as the letter {p} in word-initial {spec-}, despite the fact that the {e} in {spec} is immediately differentiated according to post-context (i.e. /E/ in the case of {(spec)ial} rather than /i/).

This choice does not, however, imply any natural, inherent phonemic unity of /p/ or the other voiceless stops. Indeed, from the point of view of the transcription engine, [p] and [p_h] might just as well be two completely separate phonemes, as in fact they could well be in some other language. Allophones of the /e/ and /E/ vowels in Italian and allomorphs of the past-tense morpheme in English dovetailed nicely with rule-based transcription. But the natural-language rule for the aspiration of English word-initial stops did not take equal pride of place among the synthesis rules that appeared 'natural' to our system. However, this may imply less about the nature of any of these rules than it does about the nature of aspiration itself.

Rules are pure text, need no compiling, and have a language-independent format. The executable code required to run these rules is merely a regular expression matcher. This engine is the same for any language and need be compiled only once. The only operation required is to update rule set according to the language to be uttered. The text-to-phones algorithm is data-dependent and can be adapted to any language by compiling the appropriate rule set. This is not a programming option, but a data-entry option.

5 Multi-language Phonetic Data

Our research gathers phones belonging to the Italian and English languages in a single phonetic data set. This enables both languages to be fully uttered without any modification of the speech synthesizer. One positive side effect of this approach is that words borrowed from a foreign language can be uttered correctly even when embedded in the text of another language. The English word "computer" appearing in italics in an Italian text, for example, could be correctly phonetized as shown below in modified X-SAMPA (with slashes representing the transition from one phone to the next). This dual synthesis capacity is due to the mixed-language nature of the rule set:

/?/H_f/h/@/m/p/h/i@/u/oU/w/d/h/3/r\/

where:

/?/ is a glottal stop
H_f is hold fricative closure
/h/ is {h} in {hand}
/@/ is {u} in {cup}
/m/ is {m} in {man}
/p/ is {p} in {ape}
/i@/ is {y} in {any}
/u/ is {oo} in {boot}
/oU/ is {oa} in {coat}
/w/ is {w} in {wage}
/d/ is {d} in {bud}
/3/ is {i} in {bird}
/r\/ is {r} in {rage}

and correctly uttered because the phonetic set also includes phones not used in Italian speech. In the extension of X-Sampa used in our model, this word would thus be transcribed:

[?H_fh@mphi@uoUwdh3r\]

reflecting the English /k@mpju:d3r\/. Note that this contrasts with the extremely common loan word *computer*, which is normally pronounced /kom:pjute4/ in Italian.

This superset of phones also includes phones for other foreign languages, such as French, German,

and Spanish, so the speech of Western languages can already be fully covered. Some of these also come into play in representing dialectal variety in English or Italian speech. Examples of such sounds include:

/y:/ in French {menu}
/y/ in German {fünf}
/J/ in Spanish {año}
/Q/ in British {clock}
/9/ in Lombard {oeuc'} (i.e. "eye").

The rule set refers to the phones to be uttered when an alphabetical string matches a rule. Such phones are language-specific, but many languages share many phones. So a multi-language phonetic data set can be defined for large linguistic areas such as Western languages.

A universal phone set can be gathered to cover most of the world's languages, so that a single speech synthesizer can be developed. This solution will overcome the problem of sampling each language for a specific text-to-speech synthesis application. Such a speech synthesizer will need only to be driven by an appropriate phonetic sequence.

6 Formant-based Speech Synthesis

Speech synthesis is the automatic generation of the waveform corresponding to the words to be uttered. Several solutions are available to do this, but the vocal-tract model was chosen because it offers flexible control and can emulate virtually any kind of voice, including artificial voices, such as those needed in cartoons or synthetic movies.

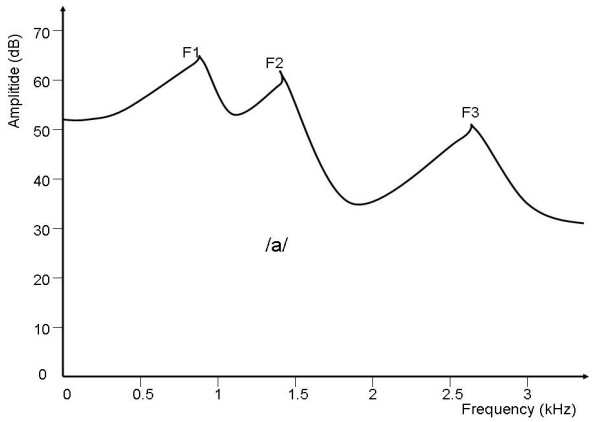

Fig. 2 – First three formants in the phoneme /a/

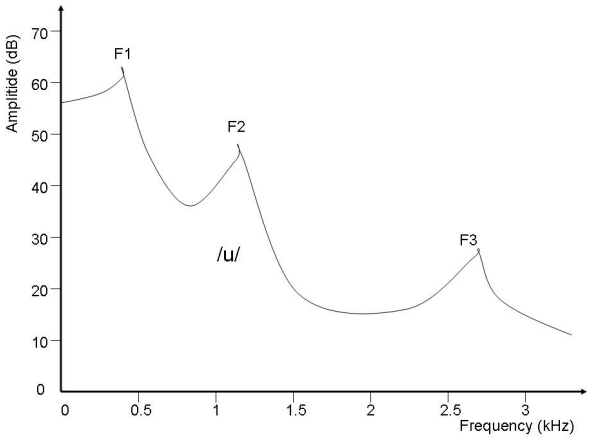

Fig. 3 – First three formants in the phoneme /u/

A formant-based speech synthesizer is an optimal model for producing synthetic utterance, because it comes very close to the human ability to produce speech. This model can be parametrically controlled in order to change speaker identity. It is also possible to produce any speech sound that an articulatory organ can make. Thus, it can cover a very large range of languages, meeting the requirements of a multilingual TTS synthesizer.

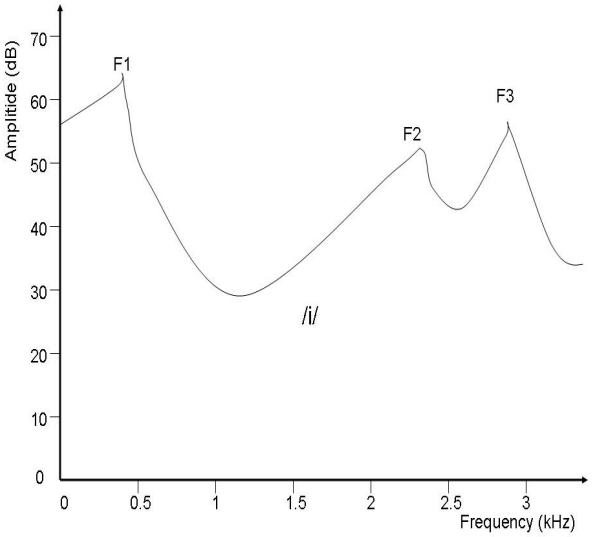

Fig. 4 – First three formants in the phoneme /i/

Formants are very representative of speech information. The first three formants enable speech intelligibility, while the others contribute to speech naturalness. This means that such a synthesis model is scalable, as required by embedded-system applications.

Comparison of figures 2, 3, and makes it obvious how the information encoded in formants practically

cries out for encoding in a representational system based on them. After all, formants are relatively easy to reproduce, and they do a very good job of encoding information in sound. Compared to the ambiguities of English spelling, the economy of this system is remarkable. Having ascertained the theoretical, acoustic validity of formant-based synthesis, the question then becomes how best to go about implementing it. Figure 5 provides a schematic model of the approach our framework incorporates.

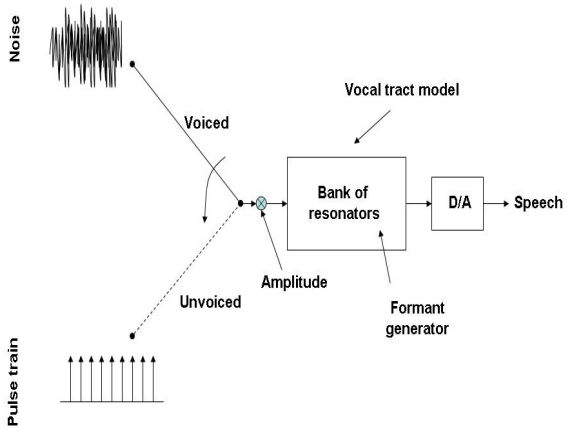

Fig. 5 – Simplified block diagram of the speech synthesizer

Several implementation models are available for formant-based speech synthesis, each with its pros and cons. The two main models are known as cascaded and parallel, terms that refer to the filter schema used to model the vocal tract. A combination of both is the best choice. A practical implementation can also be attained with a parallel model.

A parallel, filter-bank-based speech-synthesizer model was used to generate all the phones required to produce utterances according to the phonetic sequences generated from the text-to-phones transcription and its related controls:

- amplitude
- pitch
- articulation
- inflection
- rate

Amplitude enables control of prosody across the words in a sentence and across the phones in each word. Amplitude control is needed to stress the vowels on which it falls (primary stress, secondary stress, and so forth).

Pitch control enables the synthesizer to change voice (female, male, adult or child), as well as to modulate the voice across a frequency range, so that singing speech can be also generated. Pitch rate is also adjusted for interrogative and affirmative sentences (rising and falling pitch).

Controlling articulation is essential to achieve naturalness in speech production when a set of phones are sequenced. This control acts on each formant so the formants change smoothly from one phone to the next in the sequence.

Inflection is a control action on pitch so that a pitch modulation is implemented according to the inflection information embedded in the word to be uttered. Pitch-rate control is the primary level of inflection control.

Speech-rate control enables utterance production at variable speed (slow, normal or fast), while preserving pitch and formant frequency. Such control is indispensable for the natural prosodic utterance of the sequence of words in a sentence.

All the above controls are generated by text-to-phones process and are applied to the synthesizer on a frame-by-frame basis. The minimum thickness is based on the shortest phone to be synthesized. Longer phones are generated as multiple durations of the basic tick. Such time-framing allows for very fine control over how complete phones are generated, since control parameters can be gradually adjusted frame-by-frame as a phone is generated.

7 Conclusion

A framework for developing a mixed-language, text-to-speech synthesizer has been defined. A set of rules has been collected for Italian and for English. A phonetic database has been built to cover all the meaningful phonetic sounds in these two languages. Using such data, a simulation model was designed using the MATLAB developing environment to verify how much functionality can be coded by means of linguistic rules. One of the main goals of this research is to reduce the dependence of the speech-synthesis process on code, thus allowing an embedded application to switch languages or substitute speakers on the fly, without any code update. Another aim is to assist in developing standards for encoding emotion into prosody.

References:

[1] X1. J..Z.Gros, Text-to-speech synthesis for embedded speech communicators, *Proceedings of the 5th WSEAS International Conference on Artificial Intelligence, Knowledge Engineering and Data Bases*, 2006, pp. 189-193.

[2] X2. Z. Orhan and Z. Gormez, The framework of the Turkish Syllable-based concatenative text-to-speech system with exceptional case handling, *WSEAS Transactions on Computers*, Vol. 7, No. 10, 2008, pp. 1525-1534.

[3] X3. A. Conkie, A robust unit selection system for speech synthesis, "Collected Papers of the 137th Meeting of the Acoustical Society of America and the 2nd Convention of the European Acoustics Association: Forum Acusticum" (Berlin, Germany, 1999) p. 978; *IEEE transactions on speech and audio processing*, 2001, vol. 9, no. 3, p. 232.

[4] X4. C. Browman, Rules for demisyllable synthesis using Lingua, a language interpreter, *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '80*, Vol. 5, 1980, pp. 561-564.

[5] X5. P.C. Bagshaw, Phonemic transcription by analogy in text-to-speech synthesis: novel word pronunciation and lexicon compression, *Computer Speech and Language*, Vol. 12, No. 2, 1998, pp. 119-142

[6] X6. M. Malcangi, NeuroFuzzy Approach to the development of a text-to-speech (TTS) synthesizer for deeply embedded applications, *Proceedings of the 14th Turkish Symposium on Artificial Intelligence and Neural Networks*, 2005.

[7] X7. G.C. Cawley and M.D. Edgington, Generalization in neural speech synthesis, *Proceedings of the Institute of Acoustics Autumn Conference*, Windemere, U.K., 1998.

[8] X8. M. Malcangi, Combining a fuzzy logic engine and a neural network to develop an embedded audio synthesizer, *Lecture Series on Computer and Computational Sciences* No. 8, T. Simos and G. Psihoyios, eds. Brill Publishing, Leiden, the Netherlands, 2007, pp. 159-162.

[9] X9. M. Malcangi, D. Frontini, Language-independent, neural network-based, text-to-phoneme conversion, *Neurocomputing*, 2009, doi: 10.1016/ j.neucom.2008.08.023.

[10] X10. S. Anantan, R.R. Tawar, G. Loganathan, and N. Sriraam, Building speech synthesis for bahasa melayu text-to-speech system, *WSEAS Transactions on Computers*, Vol. 2, No. 3, 2003.

[11] X11. Y. March and R.I. Damper, A multi-strategy approach to improving pronunciation by analogy, *Computational Linguistics*, Vol. 6, No. 2, 2000, pp. 195-219.

[12] X12. D. O'Shaughnessy, *Speech Communication: Human and Machine*, Addison-Wesley, 1987.

[13] X13. J. Schroeter, A. Conkie, A. Syrdal, M. Beutnagel, M. Jilka, V. Strom, Y. Kim, H. Kang, and David Kapilow, A perspective on the next challenges for TTS research, *IEEE 2002 Workshop on Speech Synthesis*, IEEE, 2002

[14] X14. M. Unluturk, K. Oguz, and C. Atay, A Comparison of Neural Networks for Real-Time Emotion Recognition from Speech Signals, *WSEAS Transactions on Signal Processing*, Vol. 5, No. 3, 2009, pp. 116-125

[15] X15. M. Malcangi, Audio Interaction with Multimedia Information, *Proceedings of the 8th WSEAS International Conference on Computational Intelligence, Man-machine Systems and Cybernetics*, 2009, pp. 196-199.

[16] X16. *International Phonetic Alphabet*, http://www.langsci.ucl.ac.uk/ipa/IPA_chart_(C)2005.pdf, retrieved 2009-12-18.

[17] X17. C. McLemore, Pronlex transcription, *Linguistic Data Consortium*, University of Pennsylvania, http://www.cis.upenn.edu/~ldc/readme_files/comlex.readme.html, 1995.

[18] X18. E. Kirshenbaum, *Representing IPA phonetics in ASCII*, http://www.kirshenbaum.net/IPA/ascii-ipa.pdf, 2001.

[19] X19. *Arpabet*, http://www.cs.cmu.edu/~laura/pages/arpabet.ps and http://www.laps.ufpa.br/aldebaro/papers/ak_arpabet01.pdf

[20] X20. J. Wells, *Computer-coding the IPA: a proposed extension of SAMPA*, http://www.phon.ucl.ac.uk/home/sampa/ipasam-x.pdf, 1995.

[21] X21. A. Faber, Phonemic segmentation as epiphenomenon: Evidence from the history of alphabetic writing, in P. Downing, S. Lima, and M. Noonan (eds.), *The Linguistics of Literacy*, John Benjamins, 1992, pp. 111-132.

[22] X22. L. Canepàri, *Manuale di Pronuncia Italiana*, seconda edizione emendata, Zanichelli, 2004.