A Framework for Mixed-language 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://dsptsrts.dico.unimi.it

Abstract: - The task of text-to-speech (TTS) synthesis usually refers to a single language and to a single speaker, concatenating short parametrically controlled speech segments by means of a rule-based algorithm. The main disadvantage of this solution is its strong language and speaker dependency. We propose a framework designed to overcome this limitation, employing a multi-language text-to-speech synthesis system. The text-to-speech synthesis framework was designed to embed phonetic and prosodic information in a set of rules. Synthesis of more than one language can easily be carried out by switching from one rule set to another. The system does not depend on phone sets recorded from an actual specific human voice. Rather, it relies on a human-like, speech-synthesis model that can generate the units needed to produce the desired utterance for a specific test string in any kind of voice (male, female, child).

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

1 Introduction

Concatenative text-to-speech synthesis (TTS) has gained widespread market acceptance in recent years 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 dependence on language and use of a predefined speaker’ voice are major limitations.

Current concatenative TTS is based on patterns for storing speech (diphones, demisyllables, etc.) and on a set of rules describing how these patterns have to be concatenated to produce 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.

Its main limitation is due to the way speech segments to be concatenated are obtained. Diphones or demisyllables are strictly dependent on language and highly redundant. Diphones or demisyllables 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. 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.

Alternative approaches [3, 4, 5, 6, 7, 8, 9] to TTS synthesis might overcome the limitations of current TTS synthesis by enabling the developer to meet next-generation product requirements. We propose a TTS framework based on phonemes, allophones and pseudo-phonemes collected from different languages and a set of language-dependent text-to-speech rules. This framework also uses a formant-based speech synthesizer to generate speech segments to be concatenated [10], so that speaker dependency can be avoided. This leads to a deeply-embeddable multi-language, speaker-independent TTS synthesis model.

2 System Framework

The system framework consists of three main components: the rule-based language model, the multi-language phonetic data, and the formant-based speech synthesizer (Fig.1).

Fig. 1 – System framework for a multi-language text-to-speech synthesizer
The rules-based language model is a collection of specific-language set of rules. Each set collects all the phonetic information needed to transform an alphabetical text into the correct utterance for a specific language.

The multi-language phonetical data is a superset of phonetical data related to all the languages that will be speeched.

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

3 Rule-based Language Model

To model a language in terms of phonetics, we chose a rule-based model. 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 embedded in an alphabetical sequence. An algorithm was developed to parse the rule set and match the most appropriate rule to a specific alphabetical string. Such algorithms run on the following classes of elements defining, in terms of regular expressions, the left and right context:

(!) | (^) | ($) | (#) | ([AEIOUY]+) | (:) | ([ˆAEIOUY]*) | (+) | ([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. The following is an example of rules (not exhaustive) to match the letter E in Italian words:

!(E)! = /2e/ Isolated \{e\} is stressed \(\text{e/}\)
!(E)$$#! = /2e/ Word-initial \{e\} followed by two consonants and by one or more word-final vowels is short stressed \(\text{e/}\)
!(E)$#! = /2e:/ Word-initial \{e\} followed by one consonant and one or more word-final vowels is long stressed \(\text{e/}\)
!(E)$# = /2E/ Word-initial \{e\} followed by one consonant and one or more pre-consonantal vowels is stressed \(\text{E/}\)
!(E)$#:! = /2E/ Word-initial \{e\} followed by one consonant, a non-vowel and a word-final vowel is stressed \(\text{E/}\)
!(E)$#! = /2E/ Word-initial \{e\} followed by one consonant and one or more pre-consonantal vowels is stressed \(\text{E/}\)
!(E)$!:! = /2E/ Word-initial \{e\} followed by one consonant, a non-vowel and a word-final vowel is stressed \(\text{E/}\)
!(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.

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 /ld/, 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 continuants that were word-final prior to {-ed} suffixation. In other words, clusters like {-tch} and {-sh} need not be specifically treated:

```plaintext
!/RUGG(ED)! = /1I/1d/
!CO(ED)! = /2E/2d/
!:,(ED)! = /2d/ (e.g. red, bed, Ted)
#:(ED)! = /1i/1d/ (e.g. studied, muddied)
:(IED)! = /1a/1I/1d/ (e.g. lied, tried)
(TED)! = /1t/11/1d/ (e.g. hated, lasted)
(DED)! = /1d/11/1d/ (e.g. aided, added)
#(ED)! = /1d/ (e.g. played, hoed)
.(ED)! = /1d/ (e.g. rubbed, hugged)
(ED)! = /1t/ (e.g. stopped, worked, itched)
```

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.

### 4 Multi-language Phonetic Data

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.

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 an Italian text, for example, will be correctly phonetized as shown below in modified XSAMPA (where dashed stings represent additional articulations outside the XSAMPA phoneme set), thanks to the mixed-language nature of the rule set:

```plaintext
/?/-HFC-/h/@/m/p/h/i@/u/oU/w/d/h/3/r/
```

where:

- `/?/` is a glottal stop
- `HFC` is an 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. This superset of phones also includes phones for other foreign languages, such as German, French, and Spanish, so the speech of Western languages can be fully covered. Examples of such sounds include:

- `/y/` in the German word *funf*
- `/y:/` in the French word *menu*

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.

5 Formant-based Speech Synthesis

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.

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

Pitch control enables the synthesizer to change voice (male, female, child), as well as to modulate it across a frequency range, so that singing speech can be also generated.

4 Conclusion

A framework for the developing a mixed-language, text-to-speech synthesizer has been defined. A set of rules has been collected for the Italian and English languages and 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 code dependence of the speech synthesis process, so that an embedded application could switch languages or speakers on-the-fly, without any code update.

References:
[4] M. Malcangi, NeuroFuzzy Approach to the development of a text-to-speech (TTS) synthesizer for deeply embedded applications, in: Proc. of the 14th Turkish Symposium on Artificial Intelligence and Neural Networks, Cesme, Turkey, 2005