TTSBOX: A MATLAB TOOLBOX FOR TEACHING TEXT-TO-SPEECH SYNTHESIS

Thierry Dutoit

FPMs - Faculté Polytechnique de Mons
TCTS Lab, MULTITEL Building
1 Avenue Copernic, B-7000 Mons Belgium

Miloš Cerňak

Institute of Informatics
Slovak Academy of Sciences
Dúbravská 9, 84507 Bratislava Slovakia

ABSTRACT

The paper presents a new toolbox for teaching TTS synthesis. TTSBOX performs the synthesis of Genglish (for "Generic English"), an imaginary language obtained by replacing English words by generic words. Genglish therefore has a rather limited lexicon, but its pronunciation maintains most of the problems encountered in natural languages. TTSBOX uses simple data-driven techniques (Bigrams, CARTs, NUUs) while trying to keep the code minimal, so as to keep it readable for students with reasonable MATLAB practice. TTSBOX was designed with the hope that it can help to increase the personal involvment of undergraduate and graduate students in their TTS courses.

1. INTRODUCTION

Text-to-Speech synthesis is a complex combination of language processing, signal processing, and computer science. Students are therefore usually introduced to it in a top-down approach, emphasizing problems to be solved and introducing solutions on paper, but with little real practice : designing a TTS takes too much time, and modifying one is usually impossible if you did not take part in its design (yet only if it was correctly documented). Apart from the FESTIVAL TTS system [1], which uses SCHEME as an interactive language for letting students play with TTS basics, no real "hands on" toolbox was available, especially for engineering students (who are most often familiar with MATLAB).

TTSBOX performs synthesis of Genglish (for "Generic English"). Genglish is defined here as "English in which all the words belonging to open classes (i.e. classes of words whose number of elements is constantly expanding : verbs, nouns, adjectives, and adjectival adverbs) are replaced by a generic substitute". Apart from these substitutions, the syntax and pronunciation of Genglish is assumed to be that of English. It will also be assumed, in order to avoid the need for a preprocessor, that Genglish has no abreviations, no arabic nor roman numbers, and no acronyms. For a deeper examination of Genglish sentences, we have created a MATLAB corpus file containing a set of 50 Genglish sentences (about 800 words) in which each word is listed with its spelling, part-of-speech category, and phonetization. This file also contains a smaller test corpus (24 sentences; about 400 words), which will be used later.

Although Genglish is lexically much simpler than English (and practically looses most of its semantics), it maintains a lot of its phonetic, syntactic, and prosodic complexity. In particular, Genglish is lexically ambiguous (even more than English), since verbs and nouns can have the same spelling (but different pronunciations). Genglish is much simpler than English, in that it is a language with a closed lexicon: the list of its words is limited, once and for all. In contrast, the set of Genglish sentences is infinite. This will make our TTS design task much simpler, while keeping one of the major challenges of the synthesis of open lexicon languages : that of naturalness.

In section 2 we outline the morphosyntactic analysis of Genglish. Section 3 describes the corpus-based phonetization by using the CART method, and sections 4 and 5 describe NUU concatenative synthesis of Genglish.

2. MORPHOSYNTACTIC ANALYSIS

It is often impossible to correctly pronounce a sequence of words in natural languages without prior knowledge of their part-of-speech, as well as of their hierarchical organization into groups, which itself also depends on the sequence of part-of-speech involved. Part-of-speech information can simply be obtained from a lexicon in many cases, but there are a large number of words (most of them frequently used), which can have distinct part-of-speech, depending the context in which they are used (think of "record", "present", "answer", or "kind", in English, and "gengle" in Genglish). Genglish contains no numbers, no abbreviations, no acronyms, no complicated proper names, and no unknown words. What is more, Genglish writers make no spelling mistakes. This makes our first Genglish module a model of simplicity : the only task it has to perform is to segment incoming sentences into tokens (words and punctuations).

The morpho-syntactic module of TTSBOX is composed of a morphological analysis module, a contextual analysis module, and a syntactic-prosodic parser.

2.1. Morphological Analysis of Genglish

Since Genglish is, by definition, a closed language, the possible part-of-speech categories of its words can advantageously be described in terms of a morphological lexicon, which provides a list of all words, associated with their possible part-of-speech categories.

```
genglish_morph_lex ={
',', {'punctuation'}
'.', {'punctuation'}
'John', {'propername'}
'and', {'coordinator'}
'are',{'auxiliary'}
'be',{'auxiliary'}
'gengle',{'verb'; 'noun'}
'gengled',{'verb';
'participle'}
'gengles',{'verb'; 'noun'}
'gengling',{'participle'}
'genglish',{'adjective'}
'gengly', {'adverb'}
'is',{'auxiliary'}
'it', {'pronoun'}
'of', {'of'}
'on', {'preposition'}
'since', {'subordinator'}
'the', {'determiner'}
'to', {'to'}
'which', {'pronoun'}
};
```

Fig. 1. The (expanded) contents of the morphological lexicon of Genglish.

Fig. 2. A possible bigram automaton for Genglish (all states are supposed to be fully connected; only a few connections are shown; the white and black rings represent the initial and final states).

We therefore analyze our Genglish corpus to derive its morphological lexicon (Fig. 1). Notice the number of elements in the second column of this lexicon results from the tags we have stored in our corpus, which was our own decision. The more categories we distinguish, the more information we will have later for phonetization and syntactic-prosodic grouping; on the other hand, the design of the contextual analysis module will be harder.

2.2. Contextual Analysis of Genglish

We used n-grams for contextual analysis of Genglish. It is straightforward to see the problem in terms of a finite state automaton. In a bigram model, it is assumed that the probability of a tag only depends on the previous tag. It is then easy to sketch a bigram, using a set of states which simply represent the part-of-speech categories considered by the grammar (one state per category). Each transition is associated with a transition probability $P(c_i|c_j)$ (from state j to state i), which is the probability for a word of category c_j to be followed by a word of category c_i. If one assumes that the vocabulary is finite (as is the case for Genglish) with L the number of elements in its vocabulary, one can define, for each state and each word in the vocabulary, a state-dependent emission probability $P(w_i|c_j)$, which represents the probability that category c_j appears as word w_i.

An example is given in Fig. 2, for a possible bigram automaton of Genglish. Emission probabilities are given in text boxes attached to states. In this particular case, a large number of emission probabilities have zero value (and are therefore not mentioned in the corresponding text boxes), since all Genglish words cannot appear with all possible part-of-speech categories. Transition probabilities are attached to arcs. As opposed to emission probabilities, most transition probabilities exist a priori.
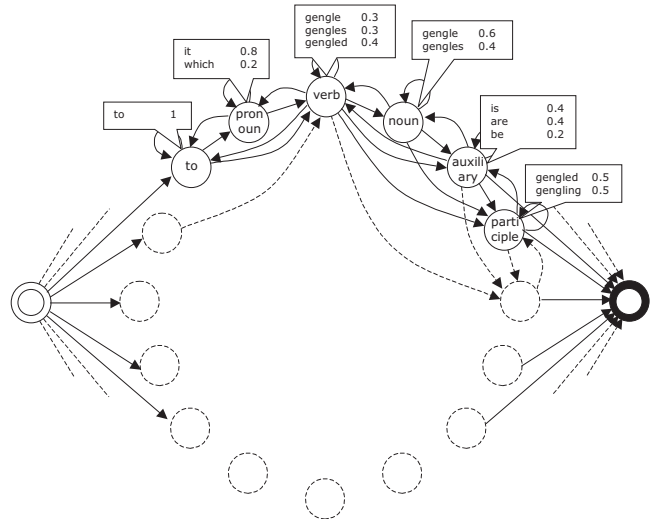
The prior computation of all emission and transition probabilities is required. This can be done by counting appearances of words and tag combinations in a corpus. The corpus must be large enough for the estimates obtained by counting to be meaningful. Fortunately, since the vocabulary of Genglish is very small, a few pages of text are sufficient. Computing bigram emission probabilities is easy in Genglish : the probability that category c_i emits word w_i is approximately given by the number of times w_i appears as c_i , divided by the total number of words with part-of-speech category c_i :

$$P(w_i|c_j) \approx \frac{\#(w_i, c_j)}{\#(c_j)}. \tag{1}$$

Similarly, the bigram transition probability between categories c_j and c_i is approximately given by the number of times c_i appears after c_j , divided by the total number of words with part-of-speech category c_j :

$$P(c_i|c_j) \approx \frac{\#(c_i, c_j)}{\#(c_j)}. \tag{2}$$

TTSBOX computes these estimates on our Genglish corpus. In practice, though, one can never be sure to cover all possible cases in a corpus, however large it is. People typically address this problem by changing zeros into small, non-zero values, which will tend to restrain the algorithm from choosing very unlikely paths, while avoiding the assumption of strict null probabilities. In our script we simply add 1e-8 to all probabilities (see [2] for more information on so-called smoothing techniques).

Once emission and transition probabilities are estimated, obtaining the best sequence of tags for a given sentence reduces to selecting the best sequence of part-of-speech tags for the sentence, i.e., the one with highest probability (given the sequence of words and the bigram model). This corresponds to finding the best path in a lattice. As a matter of
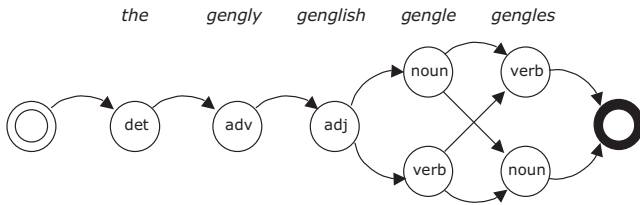
Fig. 3. An example of a lattice bigram for a simple Genglish sentence. Transition probabilities are associated with arcs. Each state is capable of emitting the word it refers to with a state-dependent emission probability.

fact, while Fig. 2 shows a bigram automaton for all possible sentences of Genglish, the automaton reduces to a lattice for a given sentence (see Fig. 3).

2.3. Syntactic-Prosodic Grouping of Genglish

In many state-of-the-art TTS systems, prosodic phrases are identified with a rather trivial chinks 'n chunks algorithm (after [3]). In this approach, we consider, a prosodic phrase break is automatically set when a word belonging to the chunks group is followed by a word classified as a chink (or, in other words, a prosodic phrase is forced to be composed of the largest possible sequence of chinks, followed by the largest possible sequence of chunks). Chinks and chunks basically correspond to function and content words classes, with some minor modifications. For the synthesis of Genglish, we consider the classes defined by Tab. 1.

Genglish chinks	"and, "since", "the", "on", "of", "to", "it", "which", "gengled" (participle), "gengling", "is", "are", "do", "does"
Genglish chunks	"gengle", "gengles", "gengled" (verb), "genglish", "gengly", "John", ",", "."

Table 1. Chinks'n chunks classes for syntactic-prosodic grouping of Genglish.

3. GENGLISH PHONETIZATION

Dealing with Genglish thus a priori offers a comfortable dictionary-based solution for phonetization. For tutorial reasons, however, we rather develop here a small corpus-based phonetizer, implemented as a decision tree trained on real data. Besides, this generic technique is increasingly used in multilingual TTS systems.

In the framework of automatic phonetization, the features used in the decision tree are simply the letter being currently phonetized, the letters on the left and right of the current letter, and the part-of-speech of speech of the current word (so as to handle heterphonic homographs). Outputs are phonemic symbols. Phonetic transcriptions are given for each word in the Genglish training corpus (the one we have used for n-grams), in such a way that each letter in the word gets its phonetic symbol (including the null symbol '_' if needed). For practical reasons, the phonetic symbols used in the corpus are chosen so that each phoneme gets a single phonetic character.

We thus create a MATLAB implementation of a CART tree (both its training and running algorithms). It is recursive, accounting for the fact that building a tree from its top is similar to buiding a tree from any of its internal nodes. We tested this phonetization on our complete Genglish test corpus (taking the part-of-speech information for each word from the corpus, not from n-gram tagging), and found no error.

4. GENGLISH PROSODY

A recent trend in corpus-based prosody generation is not to compute F0 or duration values at all. In this case, prosody is obtained as a by-product of unit selection from a large speech corpus. Thera are used phonetic features (such as current and neighbouring phonemes), as well as linguistic features (such as stress, position of the phoneme within its word, position of the word within its prosodic phrase, position of the prosodic phrase within the sentence, part-of-speech tag of the current word, etc.) to find a sequence of speech segments (or units) taken from the speech corpus, whose features most closely match the features of the speech unit to be synthesized. This is the approach we have followed in the TTSBOX. More on this in section 5.

5. CONCATENATIVE SYNTHESIS

TTSBOX also contains a simple small but efficient unit selection-based Genglish synthesizer. In order to make things simple, we have recorded the same 50 sentences as those stored in our Genglish (text) corpus and stored them in files named 1.wav, 2.wav, ..., and 50.wav.

Segmenting speech into phonemes is not an easy task, even when phonemes are known (in which case this operation is termed as alignment). Done by hand, it takes forever; done by machines, it is never completely reliable. We have used an HMM-based text-to-speech alignment system developed at TCTS Lab [4] and produced corresponding .seg files, the content of which is easy to understand: each line mentions a start, an end (in sample), and a phoneme name. Alignment, however, is conditioned by the degree of correspondence between the assumed phonemic transcription (sometimes called the orthepic transcription) and the actual list of phonetic units produced. The gap between these two worlds is undoubtedly the most difficult to bridge. Differences are mostly caused by coarticulation, which cannot be taken into account in phonemic transcriptions. Phonemic-phonetic mismatches also result from personal or local speaking styles, as mostly obvious in the way speakers set pauses in their speech (for obtaining meaningful .seg files, we have inserted pauses in the phonetic transcription of sentences after listening to the recordings). To a larger extent, performing text-to-speech alignment raises the issue of the phonetic set to use. In our segmentation, for instance, we have decided to transcribe 'gengled' into /JENgld/ and not /JENg@ld/, so as to avoid having to handle very short [@] "phonetic" units. In order to check the resulting segmentation and correct if when necessary, we have used the

WAVESURFER tool developed at the Centre for Speech Technology (CTT) at KTH in Stockholm (which reads the same .seg files).

From this segmented speech corpus, we have built a speech unit database, in which we have stored, for each available unit, the minimum information needed to compute its match to a given phonemic target. They are :

- Its phoneme,
- previous phoneme and next phoneme,
- the index of the the part-of-speech (pos) of the current word,
- the index of the current prosodic phrase (within the current sentence),
- the number of prosodic phrases on the right (until the end of the sentence),
- the index of the current word (within the current prosodic phrase),
- the number of words on the right (until the end of the current prosodic phrase),
- the index of the sentence containing the phoneme (related wav file names are given by this index),
- and the start and the end sample for the current phoneme in the related wav file.

The first member of the unit data (a string in our case), termed as the linguistic context features of the unit, does not explicitly refer to prosody : neither stress nor target tones, nor even target intonation or duration values are used here. This choice was deliberately made for keeping our unit selection as simple as possible. It even makes sense for the synthesis of natural languages, as shown by the recently developed LiONS TTS system [5].

NUU synthesis block of the TTSBOX then queries this database. First, it naturally formats targets in the same form as units (assigning them the same linguistic context feature set), except the acoustic information (in our case, the name of the wav file and the start/end sample) is missing : this is precisely what we are looking for. Secondly it checks for available diphones in the speech unit database matching the target diphones, it prunes the list down to a maximum of 10 units per diphone (so as to accelerate the search). Finally it implements a Viterbi algorithm for finding the best sequence of units, the one that minimizes an overall selection cost. The target cost is defined in a very crude way : it is 1 if the linguistic context features of unit and target match, and 0 otherwise. In other words, we have given the same weight to all linguistic features. The concatenation cost is simply set to 0 for consecutive units, and to 1 for others. No acoustic distance is computed.

Last but not least, we use a simple concatenation, which extracts selected diphones from the speech corpus and assembles them into synthetic speech. It is well known that this operation tends to produce audible mismatches (namely, spectral amplitude mismatches, pitch mismatches, and phase mismatches; see [6], Chap. 10). A minimalist treatment of phase mismatches is implemented here, by computing the cross-correlation between units to assemble, and slightly shifting the second one accordingly (so as to maximize cross-correlation around the concatenation point). Taking care of the other types of mismatches would require the more advanced frequency-domain synthesis systems.

6. FUTURE WORK

While most of the concepts used in the TTSBOX have only been used in a crude way, many refinements could be adopted to face these new issues. Let us mention, for instance: how to handle numbers, acronyms, abbreviations, uppercase titles; how to handle out-of vocabulary words, spelling mistakes; more elaborate models of prosodic phrasing should by used; the issue of corpus design should by considered; how to assess synthetic speech quality (both in terms of naturalness and intelligibility) [7].

The simplicity of TTSBOX precisely makes it possible for students not only to analyze TTS in a step-by-step way, but also and more interestingly to refine it themselves, starting from a working tool. TTSBOX is available for free at http://tcts.fpms.ac.be/projects/ttsbox/.

7. ACKNOWLEDGEMENTS

8. REFERENCES

[1] A.W. BLACK and P. TAYLOR, "Festival speech synthesis system : system documentation," Tech. Rep. HCRC/TR-83, Human Communication Research Centre, 1997.

[2] S.F. CHEN and J.T. GOODMAN, "An Empirical Study of Smoothing Techniques for Language Modeling," Tech. Rep. TR-10-98, Computer Science Group, Harvard University, 1998.

[3] M.J. LIBERMAN and K.W. CHURCH, "Text Analysis and Word Pronunciation in Text-to-Speech Synthesis," in *Advances in Speech Signal Processing*, S. Furui and M.M. Sondhi, Eds., pp. 791–831. Dekker, New York, 1992.

[4] F. MALFRERE, O. DEROO, T. DUTOIT, and C. RIS, "Phonetic Alignement : Speech-Synthesis-based versus Viterbi-based," *Speech Communication*, vol. 40, no. 4, pp. 503–517, 2003.

[5] R. BEAUFORT and V. COLOTTE, "Synthèse Vocale par Sélection Linguistiquement Orientée d'unités Non-uniformes : LiONS," in *Proceedings of JEP 2004*, 2004.

[6] T. DUTOIT, *An Introduction to Text-to-Speech Synthesis*, Kluwer Academic Publishers, Dordrecht, 1997.

[7] O. BOEFFARD and C. D'ALESSANDRO, "Synthèse de la Parole," in *Analyse, Synthèse et Codage de la Parole*, J. Mariani, Ed. Hermes, Lavoisier, Paris, 2002.