Speech Processing

Assignment 1

– TTS & Festival and Mini Literature Review

B117474

18/10/2017

Outline

Part 1: Lab report

Part 2: Mini literature review

Part1:Lab Report

1 Introduction

This report discusses theories of text-to-speech(TTS) based on practical of the Festival Speech Synthesis System. The core procedures in TTS – Text processing, pronunciation and prosody, and waveform generation, are discussed first. In light of these concepts, the pipeline of Festival(Version 2.4.0) is then explained. Through practicals, different categories of mistake in English are found in Festival. Regarding these mistakes, potential reasons are explained and possible solutions are proposed.

2 Background

2.1 TTS

TTS, also known as speech synthesis, is the process of converting text into speech, in other words, acoustic waveforms [1, p.249]. In common TTS models, there are essentially two components: a text-analysis system and a speech-synthesis system [15, p.38]. Input text is analysed into phonemic internal representation, and converted into a waveform.

2.1.1 Text processing

In TTS, text processing, or text normalisation, is “transforming an input text into a normalised form internal to the system” [4]. Text processing does not require higher-level understanding of text. Its main steps include tokenisation, non-standard words(NSWs) recognition and homograph disambiguation.

In tokenisation, text is first separated by punctuation into sentences. Each sentence then becomes a list of tokens split by whitespace. Separation by punctuation performs well mostly, except for some special cases. For example, when a period is part of an abbreviation(e.g. Prof.), period disambiguation is necessary.

Disambiguation methods are applied in sentence tokenisation. The simplest one is to implement a sequence of regular expressions in an FST [11, p.104]. Apart from the deterministic algorithm in such kind of hand-built FST-based tokenisers, there are other fancier algorithms trained by machine learning(ML) methods. They require a training set with hand-labelled sentence boundaries, and apply supervised ML methods, e.g. decision trees, logistics regression, etc. [11, p.285]. Besides, a classifier deciding sentence boundary is also useful, given successfully-extracted features of sentence boundary.

Next challenge in text normalisation is NSW. NSWs are non-standard words that could not been tokenised as normal words. NSWs come in various forms:

abbreviation(St.), acronym(UNESCO), numbers(date/month/year), etc. NSWs usually need to be expanded into words, so that they could be correctly pronounced. [14] proposes classification of NSWs and corresponding ways of expansion. NSWs can be detected by dictionaries as they normally are not in dictionaries; even some do exist, dictionaries should include ways of expansion. NSWs can be classified by hand-written rules of regular expressions or using the information from neighbouring words. For example, a four-digit number is classified as NYER(year) when preceding or following tokens of month, and expanded into two pair of digits pronounced as an integer.

The third step is homograph disambiguation, where words in the same form are pronounced differently according to context. [16] describes seven major categories of homographs and introduces three approaches of homograph disambiguation: N-gram taggers [7][15][5], Bayesian classifiers [8] and decision trees [10][13]. [16] integrates their strengths into a hybrid algorithm. The algorithm sets a k-word-width window of the target word to extract effective collocations for discrimination. For instance, it is more likely for “bass” to be pronounced as /beis/ rather than /bæs/ if preceding “player”. The width of the window could be enlarged in both directions to extract neighbouring information about the target word more than collocations, e.g. POS, trigrams, etc. All the information, as evidence for homograph disambiguation, is then computed for log-likelihoods in conditions of all possible pronunciation of the target word, with smoothing methods. These likelihoods are sorted in a descending order in a decision list for each word. Finally, the pronunciation of the target word is decided by identifying the highest line in its decision list that matches the exact context.

2.1.2 Pronunciation & prosody

The task in pronunciation and prosody is to convert the output of text processing – “discrete, linguistics, word-based representation into a continuous acoustic waveform” [15, p.192].

To decide word pronunciation, if a word exists in a lexicon with pronunciation, e.g. CMU Pronouncing Dictionary (CMU, 1993), its pronunciation is decidable; otherwise, a grapheme-to-phoneme(G2P, also known as letter-to-sound, LTS) algorithm is implemented to syllabise the unknown part of the words. Besides hand-written LTS rules, we have automatically-constructed rules by Classification and Regression Tree(CART). In CART, we search for the most probable phone sequence P, given a letter sequence L [11, p.293]:

$$\hat{P} = \underset{P}{\text{argmax}}\, P(P|L)$$

With a training set of hand-labelled letter-to-phone alignment, we compute the probability of a phone p given a letter l, applying MLE:

$$P(p_i|l_j) = \frac{\text{count}(p_i, l_j)}{\text{count}(l_j)}$$

Finally, through Viterbi algorithm, a single good alignment for each pair (P, L) is obtained, which is later used to train an ML classifier.

Prosody modification is to determine phrase break, F0 and duration. Phrase break is predicted through CART asking yes-or-no-questions about features such as length, neighbouring POS and punctuation [11, p.297]. Diphone synthesis needs F0 and duration modification. Duration is predicted by rule-based or statistical models associated with multiple duration-related factors. F0 is predicted by specifying F0 target points for each pitch accent and boundary tone, and creating F0 contour for the whole text by interpolating among these targets [1].

2.1.3 Waveform generation

In concatenative TTS system, pre-recorded units of speech are concatenated. Waveform is generated through diphone synthesis or unit selection synthesis.

A diphone is a unit consisting the latter half of the first phone and the first half of the second phone. Diphone synthesis brings diphones together. But in unit selection synthesis, instead of merely diphones, it stores units of various sizes, which enables us to select consecutive phones in database that makes the pronunciation more natural.

There are two ways of signal processing to manipulate waveform. One is Time-domain Pitch-Synchronous OverLap-and-Add(TD-PSOLA) which modifies F0 and duration through extracting frames, manipulating and recombining them by adding up the overlapped signals [11, p.309]. To increase the F0, we move the periods extracted from the original waveform closer, add the overlap and duplicate periods to keep the duration unchanged; conversely, periods are moved apart. Another one is Linear Prediction Coding(LPC). Rather than constructing waveform database, it converts waveform into parameters of source-filter models (F0 + voicing decision as source; LPC coefficients as filter). With source-filter models, LPC manipulates source to obtain desired duration and F0 and inserts filter coefficients to smooth around the concatenation points.

2.2 Festival: linking practice with theory

Festival is a concatenative TTS system and pipelined model of several modules that each “performs one specific task” [15, p.39]. Figure 1 displays its pipeline.

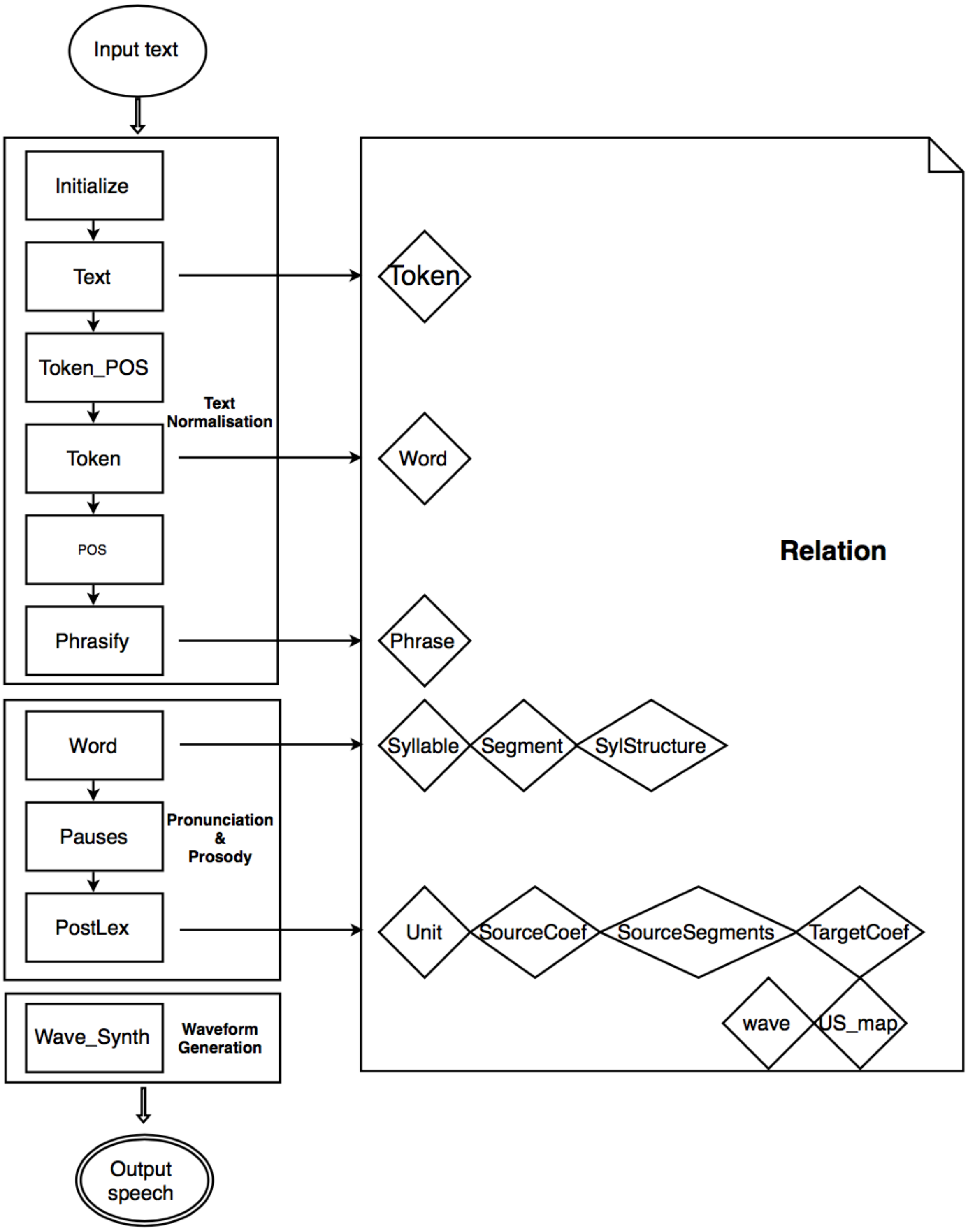

Figure 1. The pipeline of Festival (rectangle: command in Festival / procedure in TTS; square: relation created or modified after each step)

3 Finding and explaining mistakes

3.1 Text normalization

“18/10/2017”

In British English, “date/month/year” is the general way to display calendar date. In “18/10/2017”, Festival mistakenly utters it as “eighteen, ten, twenty seventeen”, as it fails in normalising it into “date/month/year”. After the Token_POS module in Festival, we see:

```
name 18/10/2017 ; whitespace "" ; prepunctuation ""
```

Festival fails to tokenise this numeric NSW. Festival uses Yarowsky-style disambiguators, each including a regular expression and a CART tree [3, 15.3.1]. For a token that matches a regular expression, a CART tree is applied to assign a class to the token [3, 15.3.1]. Back to the example, it does not match any built-in regular expression, let alone being assigned POS tags by CART. One possible solution to this problem is to add customised regular expressions to Festival’s regex-pool. Figure 2 shows one feasible regular expressions of solution in the form of FSA.

Applying this FSA, Festival should correctly tokenise the expression by slash into three parts and give POS tags which are also the NSW categories: “18” – day, “10” – month, “2017” – year. They then could be expanded into correct words: eighteenth, October, two thousand and seventeen/twenty seventeen.

Festival does not perform well neither in several other forms of calendar dates, e.g. “18.10.2017”, “18/Oct/2017” and “18-Oct-2017”. Correct tokenisation could be realised through simple modification in Figure 2, e.g. adding dot and hyphen with slash, adding abbreviation of month, etc.

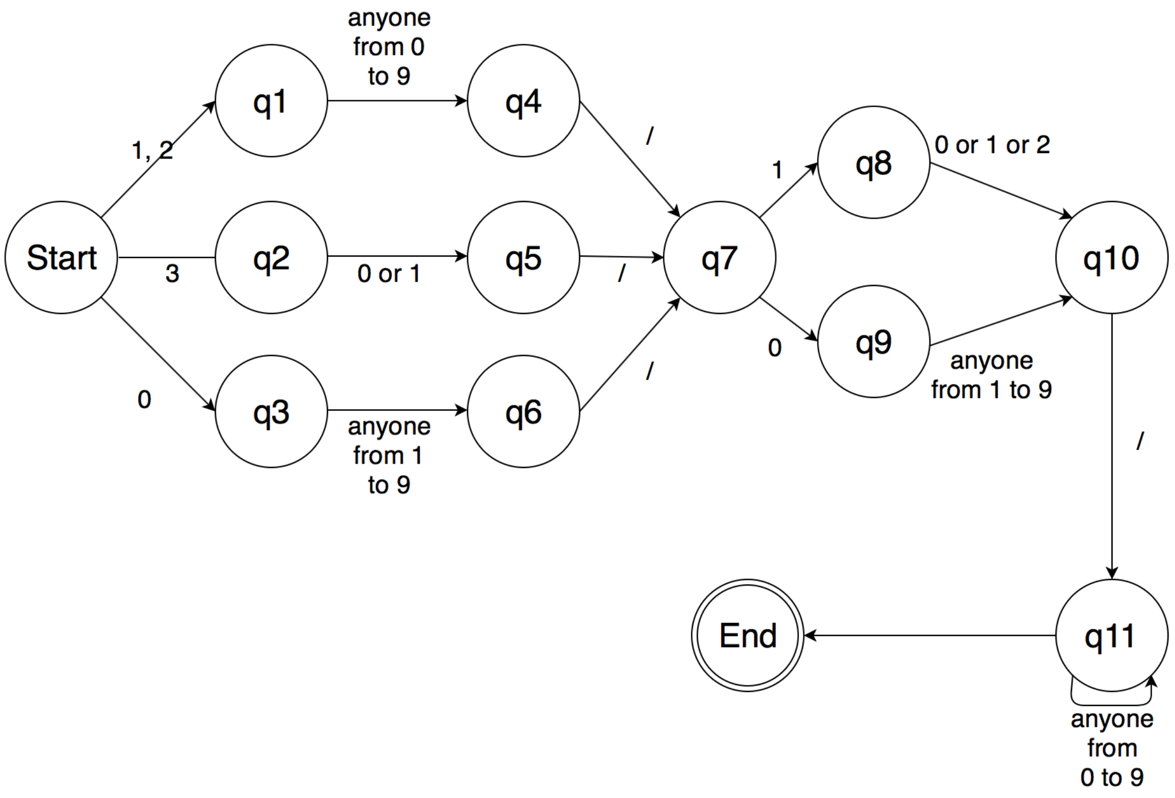

Figure 2. A feasible regular expression in the form of FSA that could correctly tokenise British-style calendar date

3.2 POS tagging

“a toy car”

The phrase “a toy car” is tagged as follows.

```
name a   ;pos dt
name toy ;pos jj
name car ;pos nn
```

Where a noun tag(nn), is expected, Festival tags “toy” as an adjective(jj). The “toy” plays the role as a noun modifier of the head noun “car”, which should not change its POS.

The same mistake occurs in “a desktop computer”(an example from [2]), “a dolphin brain”, etc.

Festival includes a HMM-type POS tagger, so POS tags are assigned based on the probability distribution of tags given a word and *N*-Grams of tags. In other words, it depends on both the transition probability of hidden states, and the emission probability of each token-tag pair. The assignment of POS tags, or say decoding, is implemented by Viterbi algorithm, a kind of dynamic programming.

In “a toy car”, the joint probability of “a toy car” and “dt jj nn” is:

$$\begin{aligned}&P(a\ toy\ car, dt, jj, nn)\\ &\quad = P(dt|<s>)P(a|dt)\cdot P(jj|dt)P(nn|jj)\cdot P(toy|jj)P(car|nn)\\ &\quad \cdot P(</s>|nn)\end{aligned}$$

And the joint probability of the correct tags for the example is:

$$\begin{aligned}&P(a\ toy\ car, dt, nn, nn)\\ &\quad = P(dt|<s>)P(a|dt)\cdot P(nn|dt)P(toy|nn)\cdot P(nn|nn)P(car|nn)\\ &\quad \cdot P(</s>|nn)\end{aligned}$$

Thinking about why Festival assigns an impossible “jj” for “toy”. From the equations above, it is reasonable to infer that $P(jj|dt)P(nn|jj)$ is larger than $P(nn|dt)P(toy|nn)$, so that Festival’s Viterbi decoder follows the path of assigning “jj” to “toy”. This might be related to the actual method of smoothing applied in the decoder, and the size of the corpus as training set for the HMM tagger model.

Assuming no smoothing is implemented in the model, $C(toy, jj)$ or $P(toy, jj)$ should be zero. To avoid zero occurrences, smoothing is implemented to the model. However, to tackle the current POS tagging problem, we could look back to the original count or probability of the word-tag pair before smoothing. If the original count or probability is zero, then the tagging result based on HMM with smoothing should be suspicious, and we should backtrack to one previous state to find a possible route.

3.3 Phrase break prediction

"… own – government …"

```
name own ;     pos jj ;  pbreak NB
name – ;       pos punc ;pbreak NB
name government ;pos nn ;  pbreak NB
```

It is natural to insert a break after an em dash when it connects two parts in a sentence to emphasise the latter part. But the Phrasify module in Festival fails to do so.

Festival utilises a CART tree and a probabilistic model to predict phrase breaks, including BB(big break), B(break), and NB(no break). From [3, 17], the default tree of phrase break is effective but simple, only including "?", ".", ":", "'", "\" and ";" as nodes. So, we can simply add an em dash as a node in the CART tree. The probabilistic model integrates an *N*-Gram model with a Viterbi decoder, and is then used to predict phrase break between words. The model assumes that "a break after a word" is based on "the part of speech of the neighbouring words and the previous word" [3, 17], and includes an *N*-Gram of distribution of breaks and non-breaks. But it does not insert any break in such an example "an old big furry beautiful Scottish hat" that has five consecutive adjectives. One possible reason might be that, as Festival claims, it typically uses a 6-or-7-Gram. This leads to extremely small and close values of probability in the model due to data sparsity. As a result, it does not perform well for this 7-Gram "dt jj jj jj jj jj nn".

3.4 Pronunciation

“selfiegenic”

“Selfiegenic”[12] is a recently-created adjective which describes someone attractive in selfies. Google returns 1,350,000 results of “selfiegenic”. Its pronunciation in Festival’s lexicion is:

```
(((s e l f) 0) ((ii) 0) ((i jh) 0) ((e n) 0) ((i k) 0)))
```

LTS rules are applied to “selfiegenic” just after the Pauses module. The wrong application of LTS rules here is assigning sound “i” to letter “e. The highlighted part in Figure 3 clearly proves this. The correct phone for “e” should be “ϵ”, indicating silence.

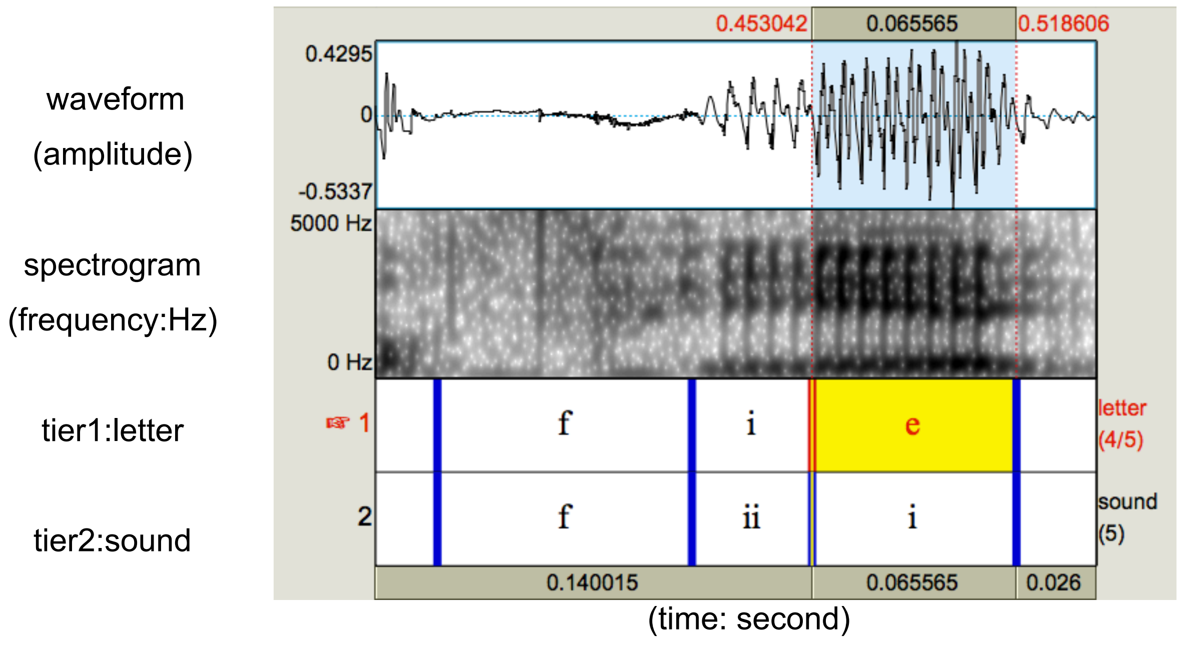

Figure 3. The “fie” part of the waveform and spectrogram of “selfiegenic” generated by Festival

CART tree is implemented in pronunciation prediction. It asks binary questions about every letter to determine their phone. These questions and their order are learned automatically based on training data containing “alignment that tells us which phones align with which letter" [11, p.294]. Then the LTS CART tree decides the best sound for each letter. Figure 4 shows part of a possible improved LTS CART tree for letter "e".

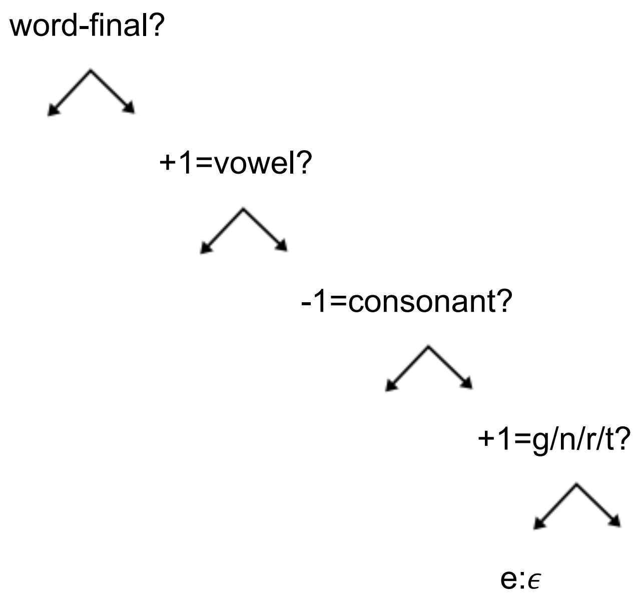

Figure 4. A possible CART tree deciding the phone for letter "e"
(left branches denote answer yes; right branches denote answer no)

3.5 Waveform generation

“discuss” and “discussion”

Voiced onset time(VOT) indicates aspiration in consonants. Aspirated sounds have longer VOT. Both “c” in “discuss” and “discussion” should be transcribed as unaspirated stop [k], which has a short VOT. Part of the waveforms and spectrograms of “discuss”(Figure 5) and “discussion”(Figure 6) by Praat, with VOT highlighted, are displayed.

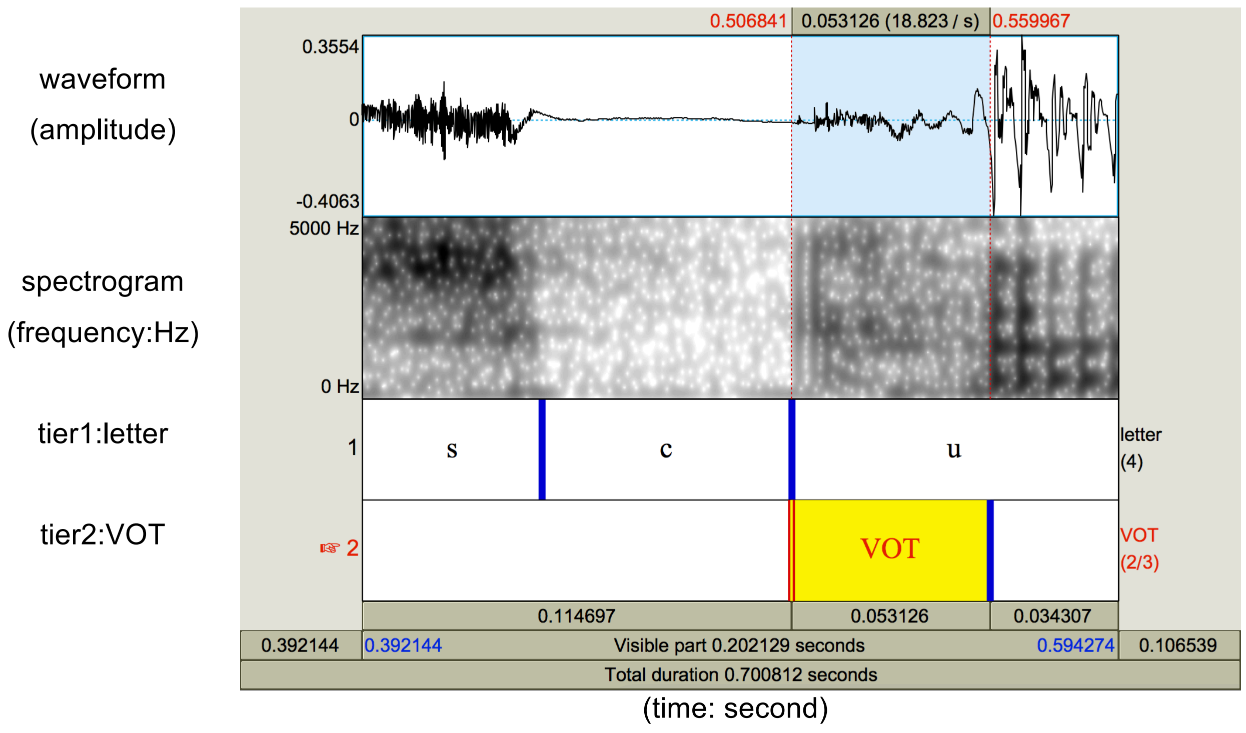

Figure 5. The “scu” part of the waveform and spectrogram of “discuss”

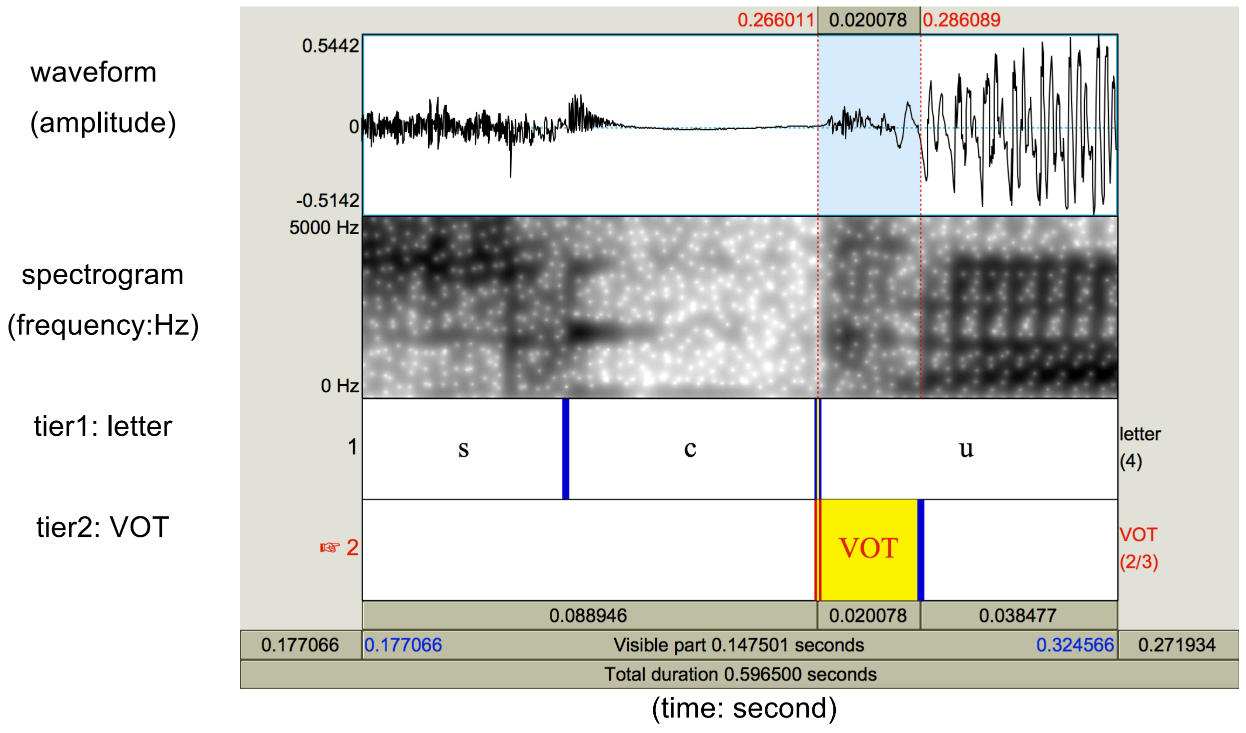

Figure 6. The “scu” part of the waveform and spectrogram of “discussion”

The VOT for the “c” in “discuss” is 0.053126s; the one in “discussion” 0.020078s. This big gap results in perceptually distinct aspiration in “discuss” and more natural pronunciation in “discussion”.

Festival employs TDPSOLA and LPC to modify F0 and duration. We could infer that for the “c” in “discuss”, its duration is either overlengthened by adding too much space between periods, or incorrectly modified by the source-filter model, so that the aspiration becomes extremely clear.

This problem might be solved by applying unit selection synthesis. We can pick up the best piece of “sc” from database without signal modification by PSOLA. Meanwhile, we could train LPC models with more data distinguishing aspiration to achieve better generation.

3.6 Other mistakes

“see Marc” vs. “seem arc”

In English, “glottalisation resolves potential word boundary ambiguity in stop before sonorant contexts” [6]. In “seem arc”, we expect to have glottalisation after “m”, which distinguishes it from “see Marc”.

Festival fails to do correct post-lexical modification to “seem arc”. They have similar segment relation after PostLex module:

“see Marc”	“seem arc”
`name s`	`name s`
`name iii`	`name ii`
`name m`	`name m`
`name ar`	`name ar`

By creating built-in glottalisation rules and modifying CART trees deciding the application of such rules are possible solutions.

4 Discussion and conclusion

Current TTS systems highly integrate a number of algorithms and models in theory to build practical modules dealing with different TTS problems. From the simplest regular expression, to CART tree, *N*-Grams, HMM, and Viterbi, they are implemented in TTS tasks including tokenisation, pronunciation prediction, waveform generation, etc. However, the systems are far from perfect. Further improvement in algorithms and models is necessary. Meanwhile, users can customise functions in the libraries of many TTS systems to meet their goals.

Bibliography (in alphabetical order)

[1] Anderson, M. J., Pierrehumbert, J. B., and Liberman, M. Y. (1984). *Improving intonational phrasing with syntactic information*. In ICASSP-84, pp. 2.8.1–2.8.4.
[2] Barker, K., & Szpakowicz, S. (1998, August). *Semi-automatic recognition of noun modifier relationships*. In Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1 (pp. 96-102). Association for Computational Linguistics.
[3] Black, A., Taylor, P., & Caley, R. (2014). *The Festival Speech Synthesis System, System Documentation*. Edition 2.4, for Festival Version 2.4.0. http://www.festvox.org/docs/manual-2.4.0/festival_toc.html (Access date: 18/10/2017)
[4] Breen, A., Eggleton, B., Dion, P., & Minnis, S. (2002). *Refocussing on the Text Normalisation Process in Text-to-Speech Systems*. In Seventh International Conference on Spoken Language Processing.
[5] B. Merialdo. *Tagging text with a probabilistic model In Proceedings of the IBM Natural Language ITL*. Paris, France, 161–172, 1990.
[6] Edgington, M. (1997). *Investigating the limitations of concatenative synthesis*. In Eurospeech.
[7] F. Jelinek. *Markov source modeling of text generation*. In Impact of Processing Techniques on Communication, J. Skwirzinski, ed. M. Nijhoff, Dordrecht, 1985.
[8] F. Mosteller & D. Wallace. *Inference and Disputed Authorship: The Federalist*. Addison-Wesley, Reading, MA, 1964.
[9] K. W. Church. *A stochastic parts program and noun phrase parser for unrestricted text*. In Proceedings of the Second Conference on Applied Natural Language Processing, 136–143, 1988.
[10] L.Brieman, J.Friedman, R.Olshen, & C.Stone. *Classification and Regression Trees*. Wadsworth & Brooks, Monterey, CA, 1984.
[11] Martin, J. H., & Jurafsky, D. (2009). *Speech and language processing.* Second Edition. New Jersey: Pearson Education.
[12] Merriam-Webster New Words & Slang. http://nws.merriam-webster.com/opendictionary/newword_display_recent.php (Access date: 21/10/2017)

[13] P. Brown, S. Della Pietra, V. Della Pietra, & R. Mercer. *Word sense disambiguation using statistical methods*. In Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics, Berkeley, 264–270, 1991.
[14] Sproat, R., Black, A. W., Chen, S., Kumar, S., Ostendorf, M., & Richards, C. (2001). *Normalization of non-standard words*. Computer speech & language, 15(3), 287-333.
[15] Taylor, P. (2009). *Text-to-speech synthesis*. Cambridge university press.
[16] Yarowsky, D. (1997). *Homograph disambiguation in text-to-speech synthesis*. In Progress in speech synthesis (pp. 157-172). Springer New York.

Part 2: Literature Review

– TTS evaluation

[1] evaluates TTS systems by asking human subjects to assess the intelligibility of synthesised sentences. The sentences are automatically-generated semantically-unpredictable ones, constructed by five simple basic syntactic structures and lexicons of mini-syllabic frequent vocabulary. Compared with evaluation at other levels, the one in [1] seems more reliable and flexible cross-linguistically.

In [2], to report the weakness of time-domain concatenative TTS systems which without explicit speech models, it designs three kinds of experiments that are beyond systems' capabilities. Consequently, some problems of TTS systems are revealed due to the bad performance. And it suggests possible solutions accordingly.

Both papers talk about evaluation methods of TTS systems with something in common. First, they both include human subjects' participation in intelligibility assessment, emotion classification, or ambiguity judgement. So, the evaluation of TTS systems is directly dependent on subjects' judgement. As TTS systems so far mainly service human beings, it is reasonable for both papers to have human feedback as measurement. Second, they both (partly in [2]) evaluate sentence-level performance of TTS systems. [1] claims sentence-level intelligibility of the first priority; [2] implements the emotional classification experiment with sentences which are long enough for subjects to perceive and judge.

Meanwhile, they differ in some aspects. First, they have different objectives. [1] aims at constructing a robust and flexible test to various TTS systems; [2] tries to prove the importance of containing an explicit speech model. Second, the systems are evaluated by different aspects of language. In [1], subjects judge how intelligible the synthesised sentences are to them; subjects in [2] classify emotions of utterances, which should not require any understanding of the sentences' actual meanings.

Both papers are imperfect. As stated by itself, the five syntactic structures in [1] used to generate sentences has no scientific support. It is suspicious whether the number or variety of syntactic structures would affect the result or not. Another unclear point in [1] is the relation between the intelligibility results of synthetic and natural utterances(Table1). Readers might wonder what the close values in each test mean. In [2], as the content of sentence usually have an influence on emotion classification, I doubt the results of the emotional synthesis experiment might have been biased. It is better to select semantically-meaningless sentences as experiment objects. Besides, [2] fails to take other factors into consideration in the database experiment. The unnaturality in a0743s05 might be resulted from the failure of TDPSOLA in modifying duration correctly, as perceptually the natural utterances are smooth in general but short and abrupt. Also, unnaturality of single words does not necessarily lead to unnaturality in longer utterances.

Relating back to Festival, when intentionally seeking mistakes it makes, we could consider more acoustically, e.g. looking at glottalisation as proposed by [2] to see whether Festival performs well in PostLex modification of glottalisation. Also, we can try synthesising semantically-anomalous sentences to analyse Festival’s performance in tasks such as POS tagging, waveform generation, etc., which should bring interesting outcomes.

Bibliography (in alphabetical order)

[1] Christian Benoît, Martine Grice & Valérie Hazan. *The SUS test: a method for the assessment of text-to-speech synthesis intelligibility using Semantically Unpredictable Sentences*. Speech Communication, vol. 18, 381-392. 1996.
[2] M. Edgington, *Investigating the Limitations of Concatenative Synthesis*. In Proc. Eurospeech. volume 2, pages 593-596. Rhodes, Greece. Sep 1997.