Speech Processing

Speech Synthesis Lab Report

Anshun Zheng

Word Count: 1897

School of Philosophy, Psychology & Language Sciences,
University of Edinburgh

8 November 2022

Contents

List of Figures

1 Introduction

Speech synthesis or TTS technology is what humans have been chasing for several centuries and has diversified applications like "read out for the blind" (Jurafsky & Martin, 2009, p. 283). This report is aiming to demonstrate the mechanism implemented in a modern TTS system–Festival, by inspecting each pipeline and its underpinning theoretical basis. The errors that Festival fails to handle on the way will also be addressed with an analysis before discussing the potential future development. Note that the voice here employed is from a scottish speaker.This dialect variation and individual difference, will also be considered in the following analysis.

2 Background

It is impossible to create a human-like voice without knowing how humans speak. There are many approaches to analyzing human communication such as neuro-biology, linguistics, and information theory, from which perspective Taylor (2009, p. 21)divided speech production into 'message generation' and 'speech encoding', where the former generates a message containing a meaning and the latter converts it to corresponding forms, say, phonemes or graphemes and also automatically adds para-linguistic information such as prosody, and physiological differences. However, TTS systems do not generate meanings themselves, instead, they decode from the written forms to retrieve phonemes and prosodic information(partly) and encode those into speech signals as shown in Figure 1.

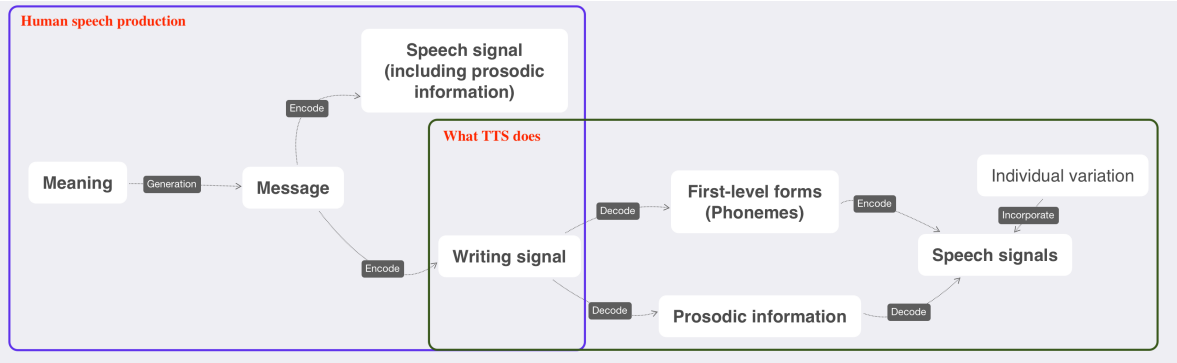

Figure 1: Comparison between human speech production and TTS system, a simplified version

The TTS system employed in Festival is a concatenative synthesis system using unit selection method(Catherine, 2022,slide6). It constructs an utterance-based architecture, where a pipeline of modules will be executed on utterances. An overview of its modules and their purposes is in Figure 2.

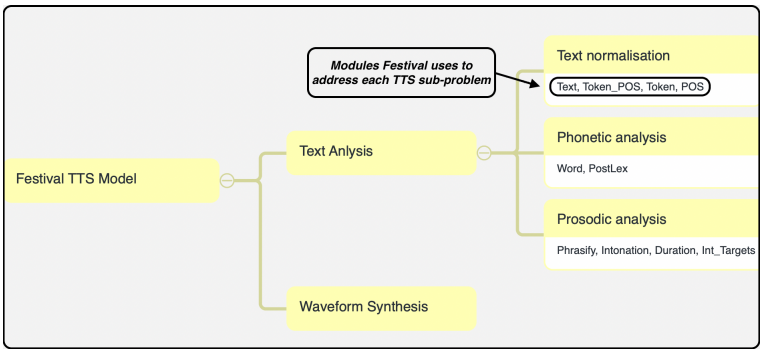

Figure 2: Modules in Festival with their purposes

The structure is based on Jurafsky and Martin, 2009, Ch.8

To begin with text analysis, the first step is text normalisation which includes sentence tokenisation and non-standard words(NSW) normalisation(Jurafsky & Martin, 2009, p. 285), to which Taylor (2009, p. 53) further added decoding and parsing (see Figure 5). There are two essential ideas in this process :first, texts should be split into tokens and then chunked into utterances(i.e., words, phrases) for such reasons as the ease of succeeding processes and helping in marking prosody, say, final-lengthening; second, we should resolve the problems brought by natural or non-natural language like NSW verbalisation and homograph disambiguation(Taylor, 2009, pp. 64, 67). In Festival, it first takes advantage of whitespaces and punctuations to tokenise the utterance and chunk them into sentences. However, simply having space as delimiters is overlooking the task's complexity(Webster & Kit, 1992, p. 1106), for example, a sentence can end by a colon and a period can be"abbreviation-final"(Jurafsky & Martin, 2009, P251). This end-of-utterance ambiguity is handled by a context-based decision tree in Festival(Alan, 2000,5.3). Then, Festival will expand NSWs using one or more finite state transducers(FST) like Figure 3 in its 'Token' module. One predictable problem here is the reading of a number is highly dependent on individual differences, for example, 'R531' can be read as 'Room five thirty one' or

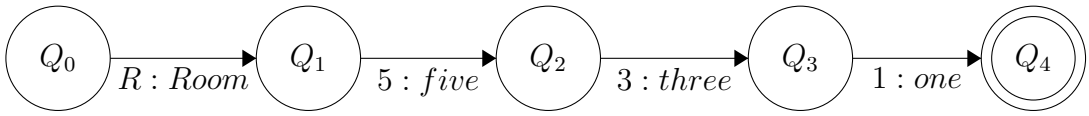

Figure 3: FST for 'R531'

'Room five hundred thirty one'(see Figure 4), and even 'r' can have other interpretations, say, radio, so more and more FSTs need creating, which may bring in more problems. Such words that share the form but differ in pronunciation are homograghs. Basically, Festival classifies them by going through

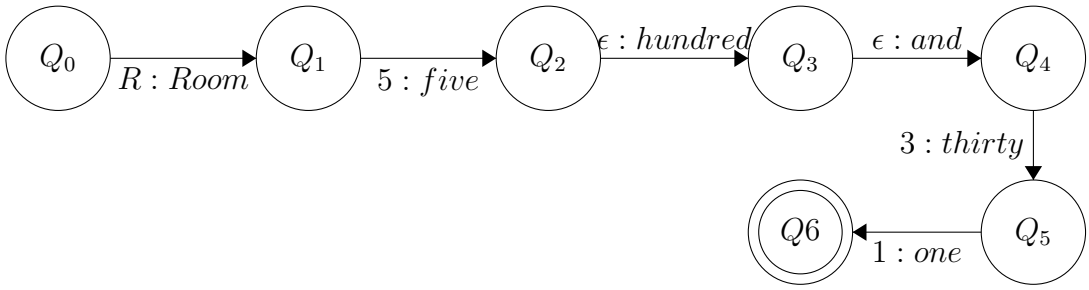

Figure 4: FST for 'R531':second reading

decision trees that are built on contextual information (i.e., five words before and after) extracted from a large corpus(Alan et al., 2014,15.3). Meanwhile, it also utilises the preliminary PoS features in Token_POS module, since some homographs like 'does' do not share the same PoS (N: female deer; V: auxiliary) and can be distinguished by PoS tagging easily.

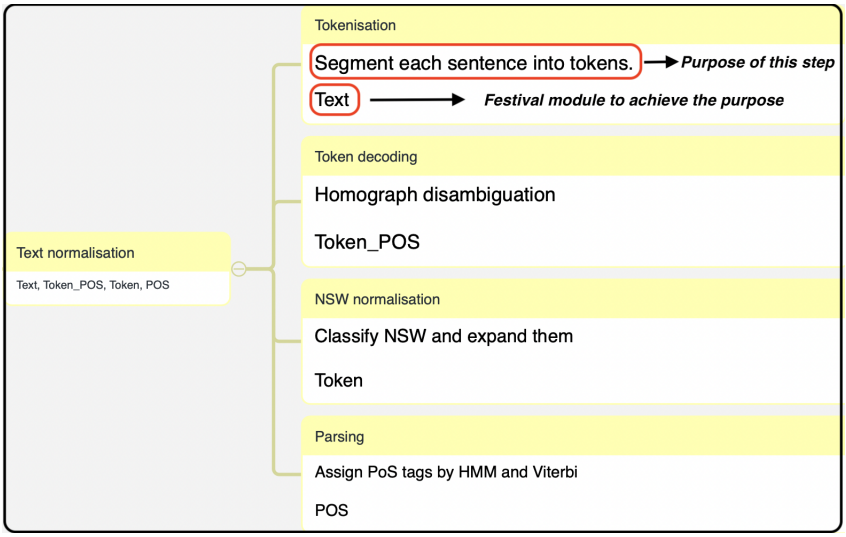

Figure 5: Text Normalisation

Basic structure from Jurafsky and Martin, 2009 but revision made based on Taylor, 2009

After tidying up the noisy writing signal, we will fill out the phonetic specification for each normalised word. Usually it is done by looking through a dictionary that contains specifications of phonemes, syllables, stress, and so forth (Jurafsky & Martin, 2009, p. 298). However hard a lexicon might try, there will be words never occurred in the dictionary, and that is where letter-to-sound(LTS) rules come in(one example see Figure 6). We want to take account both phonetic fine-grained details and possible phonological rules, so this task is highly context-sensitive. Nonetheless, Festival has loaded a compiled

$$g \rightarrow \begin{cases} /j/ \,/\, \_e,i,y \\ /g/ \quad \text{elsewhere} \end{cases}$$

Figure 6: LTS rule for letter 'g'

dictionary, a general LTS rule system and even a small list of task-specific entries(Alan, 2000, 6.1). Inside its 'Word' module, it first searches the lexicon and then transcribes the unknown words with

a “trainable LTS system”, which finds best letter-to-phone alignment and then constructs a CART model for predicting phones from the context (Alan, 2000, 6.2). An example to find the best phone for ‘g’ in ‘fringe’ is shown below (Figure 7). Note that CART in Festival will inspect the three letters before and after the target, not just one as in Figure 7, and a decision is made based on the data calculated on the training set (i.e., how many percent of ‘g’ is pronounced as/j/ before an ‘e’ versus that of /g/).

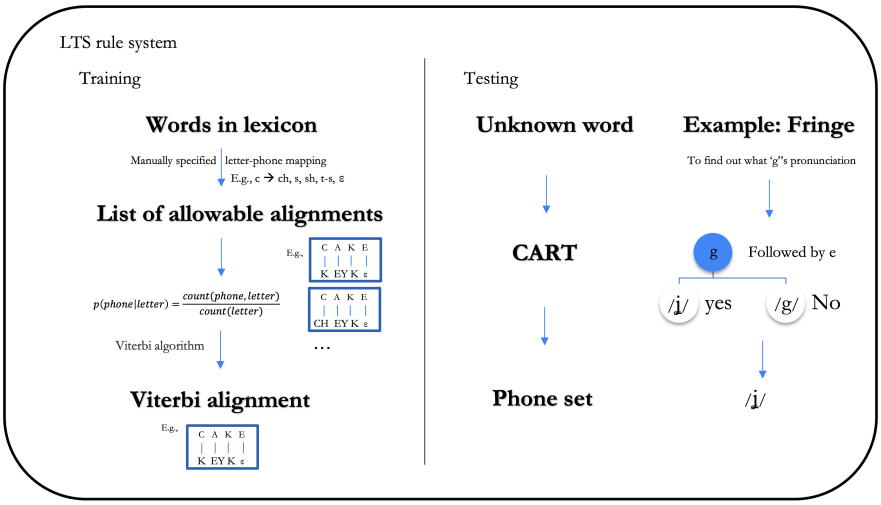

Figure 7: LTS system to find pronunciation of ‘g’ in ‘fringe’

Based on Jurafsky and Martin, 2009, p.294 and Alan, 2000, 6.2

Apart from the segmental features decoded hitherto, we need to further add suprasegmental information like prosody to imitate real speech. TTS systems will often focus on “prosodic structure”, “prosodic prominence” and “tune”(Jurafsky & Martin, 2009, p. 296). The three tasks, in another word, are finding the inter-and-intra-sentence breaks, marking both lexical-level and sentence-level stress and reflecting Fo rise and fall. Usually prosodic phrases are identified by punctuations, where always have pauses. However, it does not suffice and luckily, to solve that, Festival uses methods like decision tree(Figure 8) or statistical model based on PoS and phrase break context(Alan, 2000, 6.3) inside its ‘Phrasify’ module. Pitch accent assignment in Festival is also done by decision trees based on

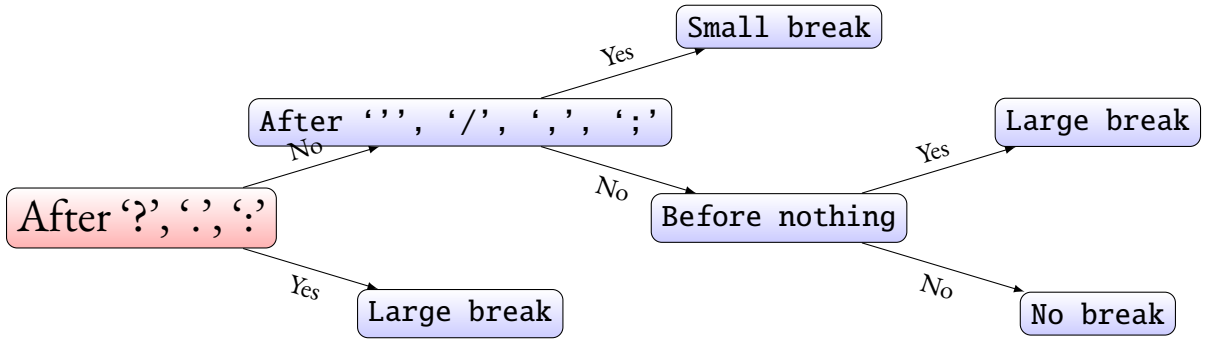

Figure 8: Decision tree for phrase break

An example from Alan, 2000, 6.3.1

different tone theories such as simple accent(Figure 9) and ToBI (Alan, 2000, 6.4.1). Different models have different accent level, for example, ToBI has 6 levels while below only marks one. Duration assignment depends on the method users use and the default setting is 100ms for every segment but this report does not deal with duration settings. Fo contour and tone assignment are quite limited in Festival but basics can be done by “rule or statistical model” (Alan, 2000, 6.7) which is beyond the scope of this report.

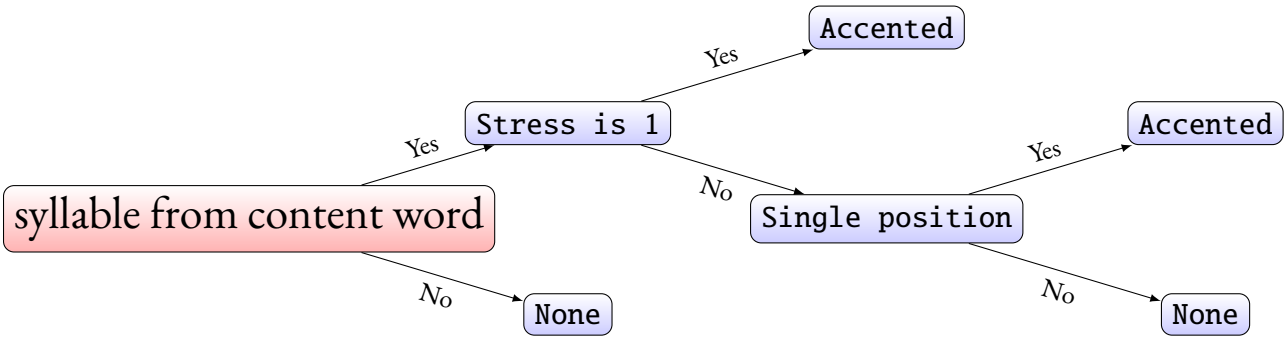

Figure 9: Decision tree for simple pitch accent

A simple example from Alan, 2000, 6.4.1

The last step of TTS is to synthesize the waveform given the specifications above.Unit selection synthesis is the method used in this report. The idea is that we select units, which can be phones of different sizes, from a naturalistic database (Jurafsky and Martin, 2009, pp.310,311, Catherine, 2022, slide 25). The unit selected should satisfy the specification of the target unit and there should be no audible breaks after concatenating(2009, p.312). In Festival’s ‘Wave_synth’ module, it will select the

best unit by finding the minimum of sum of target cost,which is the 'distance' between selected unit s_t and target unit u_t, and join cost–how well they match at the edges(Catherine, 2022, slide32-34).

$$\hat{U} = \arg\min_{U} \sum_{t=1}^{T} T(s_t, u_t) + \sum_{t=1}^{T} J(u_t, u_{t+1}) \tag{1}$$

After selecting 'best' units, they will be concatenated where amplitude is 0 (zero-crossing) but will not be adjusted by TD-PSOLA in terms of pitch and duration as in diphone synthesis.

3 Finding and explaining mistakes

3.1 Text normalisation

Input:

```
("On the 9 Sep. 2022", "exchange rete of pound" , "7.3-9.6%", "the Gov.",
↪  "£20m")
```

Output:

```
"On the ninth Sep twenty twenty two", "exchange rete of pound", "seven
↪  point three nine point six percent", "the Gov", "* * two zero m"
```

(* in the output is a unicode block)

The problems and possible causes are as follows:

1. Gov. was not expanded:

The reason it was not expanded is probably that the transducer for 'Gov.' does not exist and we can solve that by adding or improving the existing transducers.

2. '-' in '7.3-9.6' was not interpreted:

Because '7.3-9.6%' was considered as one token, which was classified as percentage. Possibly, inside the percentage transducer, it only considered numbers and dots before '%' and eliminated all other symbols.

3. '£20m':

Festival is capable to expand '£20', however when 'm' was added, the item was probably not classified as money anymore, that is, it could not disambiguate 'm' as million. Subsequently, if we disambiguate 'm' for it, it would expand '£20 pounds' into 'twenty million pounds pounds'. One way to solve it is to improve the regression model in NSW expansion by adding surrounding letters to the feature template.

3.2 POS tagging

Input:

```
"She has married a little shining reddish carrot"
"Does in the forest are beautiful"
```

Output:

```
Assigned PoS: "{prp, vbz, vbn, dt, jj, vbg, nn, nn}
{vbz, in, dt, nn, vbp, jj}"
```

In the first utterance, we found 'reddish' was mistagged as 'nn' instead of adjective. This is abnormal since Festival use HMM and Viterbi to tag. If the emission probabily $P(reddish|nn)$ is zero, which it should be, then Viterbi will not tag 'nn' to reddish. Why this happened is that there are wrong taggings in the database or smoothing was applied. Another possible reason is that the transition probability $P(nn|vbg)$ is often higher than $P(jj|vbg)$.

In the second case, 'does' was mistagged as 'vbz(third singular present)' whereas it should be 'nns', which led to wrong pronunciation. This perhaps is because, in the emission probabilities, $P(does|vbz)$ is higher than $P(does|nn)$. However, in transition probabilities, $P(vbz| < s >)$ is usually lower than $P(nn| < s >)$, counterbalancing the emission probabilities.

3.3 Phrase break prediction

Input:

```
"He...doesn't know what to do. He cried."
```

Output:

```
Predicted breaks: "{do:BB, cried:BB}"
```

The problem with break prediction is quite obvious since it only had stops after certain punctuations (i.e., '.', '.') and I assume it just simply used the decision tree in Figure 8. Normally, we should also have pauses before and after ellipsis while Festival even did not recognize the ellipsis at the stage of tokenisation. Moreover, according to Krivokapić (2007, pp. 163, 164), syntactic structure and phrase length also influence the pause placement. Therefore, an further improvement can be using statistical model to calculate the probablities of pausing in syntactic structures and attach this information when parsing.

3.4 Pronunciation

Input:

```
("It is necessary to keep chillaxing")
```

Output:

```
"{i? iz nes@s@rtˆii t@ kiip chiaksin}"
```

In the above sentence, 'chillaxing' was incorrectly transcribed. The right transcriptions should be chilaksin. The missing 'l' happened possibly because decision tree treats double 'l' environment as 'darkened l' like in 'chill' without further checking the following environment. Therefore, expanding the decision tree's sliding window will be one of the solutions.

3.5 Waveform generation

Input:

```
("We'll deliver it as soon as possible, but this may take a little
↪  longer.")
```

Output:

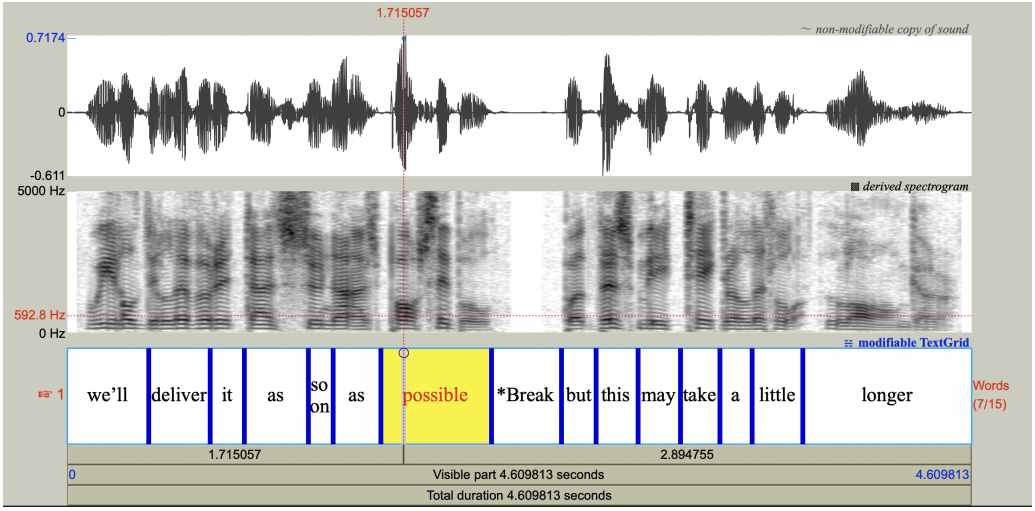

Figure 10: Annotated waveform of the input sentence

The overall wave does not sound very natural and the word 'possible' is the main cause so we will take a specific look at it (see Figure 11). At around 1.73s, we can see an obvious declination in amplitude due to concatenating two units of different amplitudes. There is also a pitch discontinuity in the middle of 'i', where an audible pitch rise happened in an unaccented position and breaks the overall pitch declination. This could be from a wrong linguistic specification, which ignored the vowel reduction in unaccented syllables.

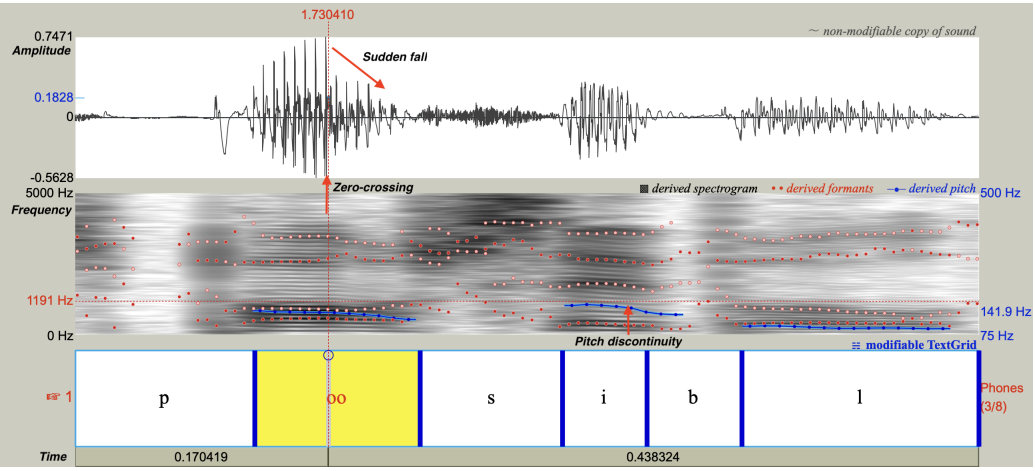

Figure 11: Waveform of 'possible' zoomed in from Figure 10

3.6 Other types of mistakes

Input:

```
"Let me live at an amazing home."
```

Output:

```
Segments: {le? miii liv a? an meizin houm}
```

Festival are preset with 'PostLex' module to further revise segments based on their contexts, however, it failed to reflect the anticipatory assimilation(lem miii). In English, the place of articulation of two neighboring consonants assimilate for the ease of speech (i.e, fat[p] man). Similarly voicing of a consonant will change because of the neighboring sounds(i.e., dogs[z]).

4 Discussion and conclusion

As an early TTS system, Festival is not perfect, whereas the errors it made can shed light on where the gap between theory and practice is and where to improve. For example, tokenisation did not identify all punctuations (i.e., ellipsis), and as a solution, we can update our default punctuation list to include all English punctuations. As a result, we may also solve the problem when predicting breaks around ellipsis. Similarly, many other mistakes had their origin in insufficient conditions either in transducers or in decision trees or in phonological rules. We may improve significantly the current system by just considering carefully those conditions. Despite the imperfectness, TTS has diversified human-machine interactions and brought voices to the disable.

Bibliography

Alan, B. W. (2000). *Speech synthesis in festival: A practical course on making computers talk* [Edition 1.4.1, for Festival Version 2.0]. Retrieved October 26, 2022, from http://festvox.org/festtut/notes/festtut_toc.html

Alan, B. W., Taylor, P., & Caley, R. (2014). *The festival speech synthesis system documentation* [Edition 2.4, for Festival Version 2.4.0]. Retrieved October 26, 2022, from http://www.festvox.org/docs/manual-2.4.0/festival_toc.html#SEC_Contents

Catherine, L. (2022). *Speech processing: Tts waveform generation* [University Lecture], University of Edinburgh.

Jurafsky, D., & Martin, J. H. (2009). *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition (2nd edition)*. Prentice-Hall, Inc.

Krivokapić, J. (2007). Prosodic planning: Effects of phrasal length and complexity on pause duration. *Journal of phonetics*, *35*(2), 162–179.

Taylor, P. (2009). *Text-to-speech synthesis*. Cambridge University Press. https://doi.org/10.1017/CBO9780511816338

Webster, J. J., & Kit, C. (1992). Tokenization as the initial phase in nlp. *Proceedings of the 14th Conference on Computational Linguistics - Volume 4*, 1106–1110. https://doi.org/10.3115/992424.992434