The University of Edinburgh

Speech Processing

Assignment 1

TEACHER: Simon King

Word count: 2482 + 465

EXAM NUMBER: **B112767**

DATE: 25 Oct 2017

Part I: Lab report

1 Introduction

University of Edinburgh's Festival Speech Synthesis system, a text-to-speech(TTS) toolkit, has been a popularly utilized free-source speech software in terms of speech synthesis research (Clark et al., 2007). Speech researchers could use Festival to read a sequence of words, complete tokenization and part-of-speech tagging tasks, and process waveforms among different languages. In the first assignment of Speech Processing course, we were guided and encouraged to find out mistakes of this system, in order to not only help understand the speech synthesis procedure but also apply theoretical knowledge into real practice by making efforts to propose some solutions of the mistakes of different categories.

2 Background

Speech synthesis, which is also called text-to-speech(TTS), means it could produce speech results once given natural language sources. Jurafsky and Martin (2014) hold the view that TTS could be divided into two parts: text processing and waveform synthesis. The procedure of text processing consists of text processing (including Sentence Tokenization, Non-Standard Words(NSWs) expanding, POS tagging, pronunciation and prosody analysis and waveform generation. Figure1 below shows the workflow of how Festival Toolkit processes the text sequences and synthesize waveforms afterwards.

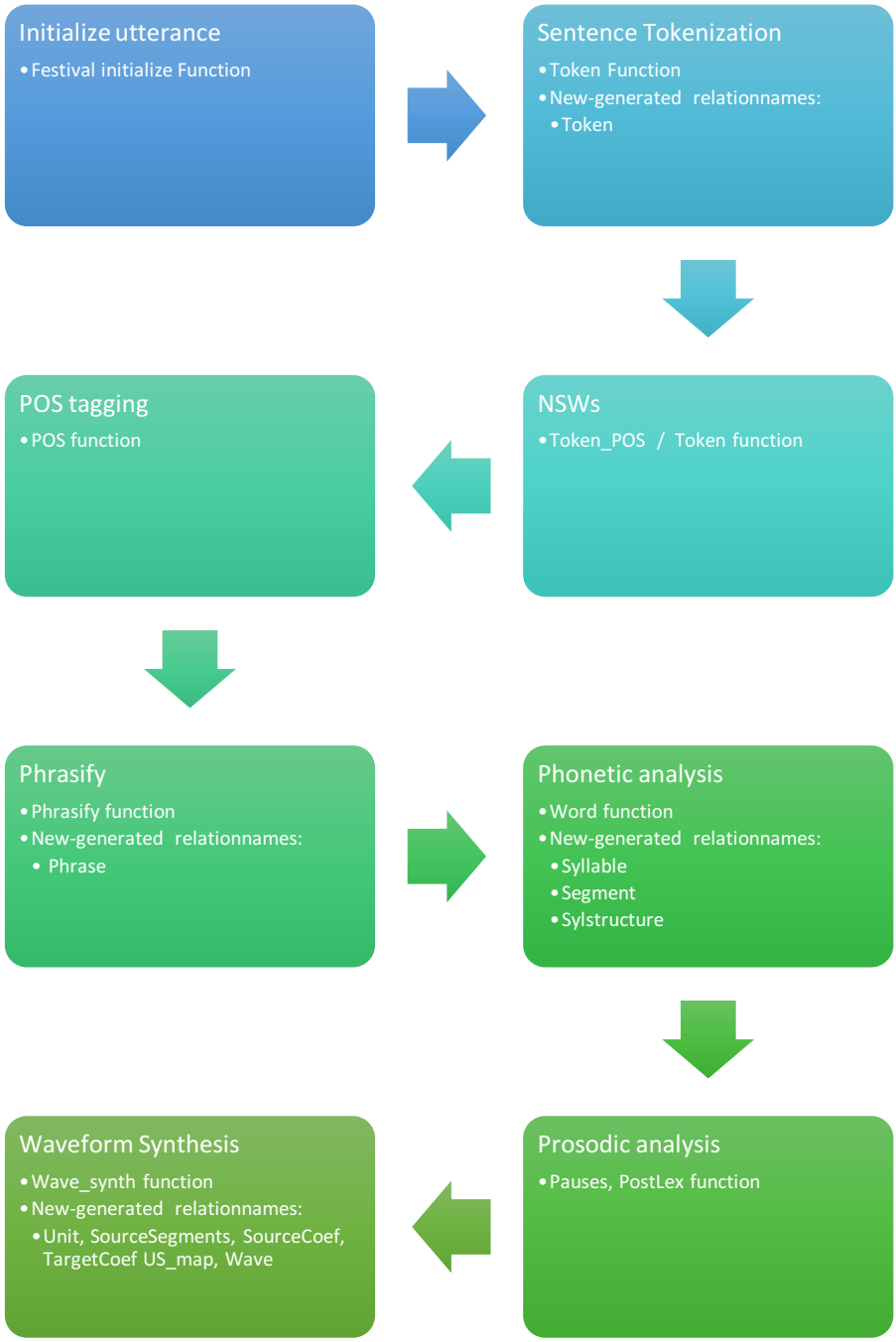

Figure1. the process of Festival Pipeline Schema

2.1 Text processing

In Festival toolkit, the word sequences should be initialized by Initialize command before starting processing. After initialization, the input sentences will be normalized, which means lexicons should be parsed, recognized and expanded as the normal

words. For example, every NSW, such as abbreviations, would be expanded into English words which can be correctly pronounced by Festival. Then Festival will do Part-Of-Speech(POS) tagging based on this. In this stage, the workflow can be divided into three parts: sentence tokenization, NSWs normalization and POS tagging.

2.1.1 Sentence Tokenization

In most occasions, tokens in every sentence can be divided with whitespace and therefore Festival can employ this rule to do sentence tokenization. However, there will be some problems such as period ambiguation (Jurafsky and Martin, 2014). This means it is impractical to distinguish whether a period is a sentence boundary or just an abbreviation symbol by hand-crafted rules. In this case, supervised machine learning (such as decision tree, logistic regression, support vector machine) method may be employed to deal with this problem. The festival may create labels for training data by hand (the structure of training data features can be seen at Table 1) and then train the data with the machine learning algorithm such as CART classification tree.

Previous word	**Next word**	**Other features**	**Word boundary?**
W1	W2	…	Yes
W3	W4	…	No
…	…	…	…

Table 1. Hand-crafted training data set for CART binary classification

In the Festival system, we could use Text command to tokenize utterance text with supervised learning model trained by the hand-labeled corpus. In this step, the relation Token was created by the Text function.

2.1.2 NSWs normalization

The next period in Festival is to parse and normalize non-standard words, which consists of abbreviations, acronyms, numbers or money symbols, etc. First of all, the standard words would be parsed and extracted, and then we just consider how to classify which NSW category the potential NSWs belong to. In this situation, it requires a hand-labeled inventory and many features to be fitted by a machine learning

classification model. Finally, NSWs could be expanded under the category label according to different expanding rules of the category. The Token_POS and Token function in Festival can implement this.

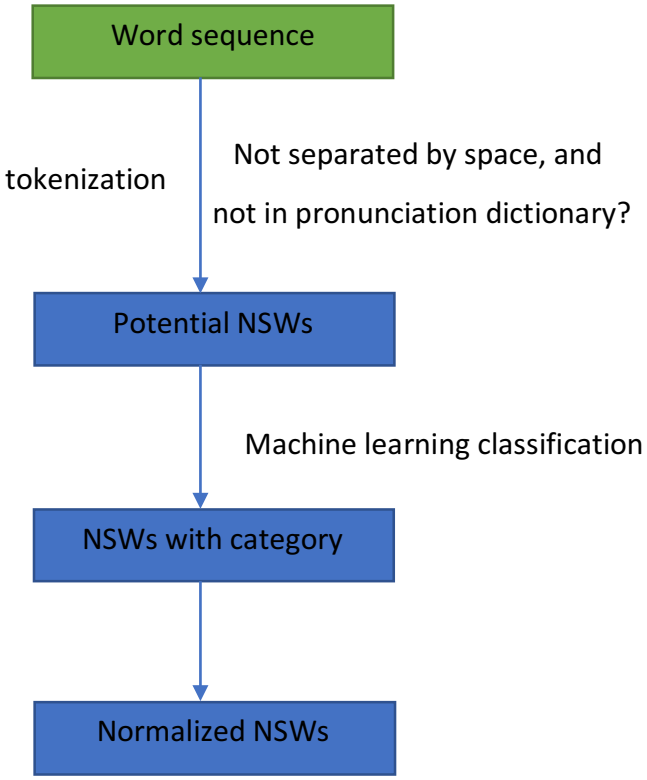

Figure 2. the workflow of NSWs normalization

2.1.3 POS tagging

Part-Of-Speech tagging is aimed to assign every token with a tag, in order to search for the appropriate pronunciation in the next phrase. In festival, we could use POS function to do this. The festival system may use the Hidden Markov Model to complete this task. In the real practice of TTS system, POS tagging is often used to deal with some homographs problems. In this step, the relation POS was created by Festival, therefore we could see the result of POS tagging using the command utt.relation.print myutt 'POS, where myutt is the variable name of our utterance.

2.2 Pronunciation & prosody

In this stage, the festival TTS system will do phonetics and prosodic analysis. Festival system would figure out the most probable pronunciation and prosody for every sentence and give a sequence of internal representation of speech to the next stage.

2.2.1 Dictionary Lookup and Phrasify

At the beginning of this stage, Festival will look up the normalized words in the pronunciation dictionary and try to find out the appropriate pronunciation. As for some words which do not exist in pronunciation dictionary, Festival would build some predictive models to make a prediction about that, which would be explained in the next stage. Festival Phrase function could be used to phrasify the string by dividing it into separate groups.

2.2.2 Letter-To-Sound(LTS) rules

As we mentioned above, after we processed the normalized word sequences including normalizing NSWs, some unseen words not in dictionary would also need to be processed. There are two methods to tackle this problem. The first method is to create a hand-labelled rules to transform certain letter sequences into phone sequences. However, it is extremely expensive and tough to deal with large and complicated corpus. Another method is to build automatic predictive model based on supervised machine learning algorithms, therefore Festival could use these well-trained models to predict the most possible speech for given text strings. Provided a string L, we could choose the pronunciation with the biggest likelihood among all the possibilities. Let P represents the probability of letter L, we could get:

$$\hat{P} = \underset{P}{\operatorname{argmax}} P(P \mid L)$$

Encountering this case, Festival system would choose the pronunciation sequence with the biggest likelihood as the equation above.

sequence	**c**	**a**	**k**	**e**
Possible phones	/s/	/a:/	/k/	/i/
	/k/	/ae/		/i:/
		/ei/		

Table 2. LTS model Predicting phones for characters

As we can see in Table2, we could computer all the possible combination of phone for every character and choose the one with the largest likelihood.

In order to obtain the best results for LTS procedure, we could not only train the machine learning classifier but could also look at the surrounding words. At some situations, the context could also affect the correct pronounce of the characters. For example, Table 3 shows two words, bit and bite. As for word *bit*, the letter *i* is pronounced by /i/. However, the letter *i* in word *bite* is pronounced /ae/. This is affected by the suffix *e*; the word *bit* does not have a final letter *e* but the word *bite* does.

string	**b**	**i**	**t**	
phone		/i/		
string	b	i	t	e
phone		/ae/		

Table 3. Comparison for letter of different context

Take the final e for example, we could build a CART classification tree for LTS system as Figure 3.

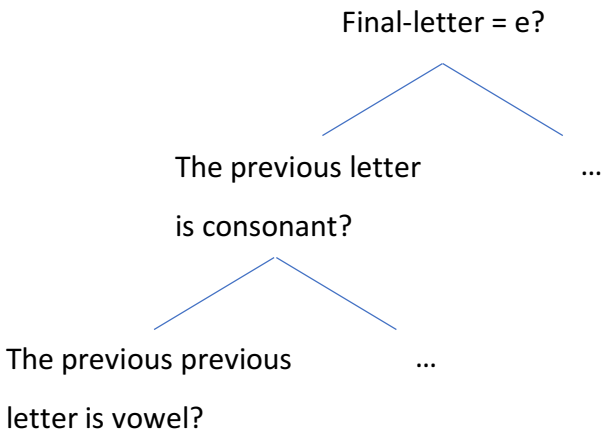

Figure 3. CART classification example for LTS rules

Taylor (2009; pp.220) claimed that it is an effective way to employ neural networks to complete this task by using feed-forward networks with a number of hidden layers. It is really a powerful method of machine learning, but we should take the computing cost into consideration because too many hidden layers and units could cost a large amount of computing resources.

2.2.3 Prosodic analysis

Afterwards, prosodic analysis would be taken. In this section, Festival should make phrase break predictions by building binary classifier based on the POS tagging results. We can create a data set by labelling all the break words and recording the previous and next several words (as Table 4). Then we could use CART classification model to build a binary classifier.

Token	Word1	Word2	Word3
Break mark	NB	NB	BB
Token	Word4	Word5	Word6
Break	NB	NB	B

Table 4. Training data for phrase break prediction

2.3 Waveform generation

After analyzing the input text, the festival system will synthesis waveform using the audio from the pre-recorded speech database. We could use Wave_synth function to complete this task, and the relation Unit, SourceCoef, TargetCoef, SourceSegment, US_map, and Wave could be generated. In this stage, Festival system concatenates different segment of diphone units and then completes some waveform manipulation tasks.

2.3.1 Diphone unit selection and concatenation

The first step is diphone unit selection and concatenation. Festival may list all the possible diphone combinations and choose the best sequence among all the candidate sequences. After choosing these sequences and concatenating them, there must be some pronunciations are not as clear as normal speech. Therefore, the festival would modify it in the next step.

2.3.2 Waveform manipulation

This step is to modify the concatenated waveform in the previous step in order to smooth the waveform and make it more likely to be as harmonious as a person's speech. One method to modify the initial waveform is TD-PSOLA, which could only

modify the F0 and its duration. Another method is using linear predicted model, which could not only modify F0 and duration but also could change the filter or even vocal tract shape (King 2017: Lecture 5, slide 4). Linear predicted model just transforms and saves the parameters of source-filter model rather than waveforms, which is space-saving and also easy to generate audio sequences.

After these two stages, Festival could generate the speech matching our input strings. We could use utt.save.wave function to save waveforms and open it in Praat. We can see as Figure 5.

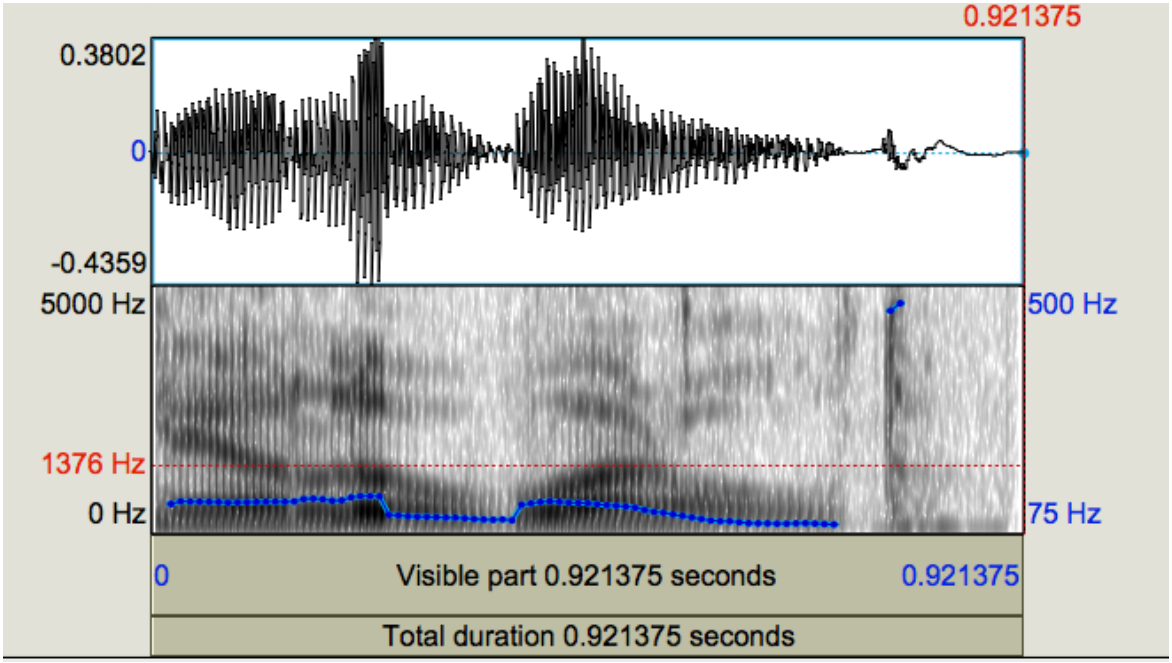

Figure 5. Waveform of utterance “Hello world”, generated by Festival

3 Finding and explaining mistakes

Although Festival is a well-developed toolkit, it could also make mistakes. Below are some mistakes I have encountered in Festival.

3.1 Text normalization

I have tried to find out the mistakes of Festival but found that in most of the time, Festival could generate speech correctly. As for some particular occasion such as the table 3 below. We could find that the movie called “Spider-Man III”(Spider-Man

Three) is pronounced wrongly by reading as “Spider-Man eye eye eye”. This minor mistake is due to the problem of NSW normalization.

Type	Text normalization
Sentences	"Spider-Man III"
mistakes	Spider-Man I I I, rather than Spider-Man

Table 5. Mistake of text normalization

Obviously, the Festival system did not train a concise model to deal with this situation. This may be due to the fact that these words seldom exist in the training corpus.
In order to solve this problem, we should enlarge the amount and richness of training data set, and create labels for this case before training supervised machine learning model.

3.2 POS tagging

Homograph disambiguation is a difficult problem in real practice. This is because words with the same spelling but different meaning and pronunciation are hard to be distinguished which tag they should belong to. In order to identify this, I have tried to input many sentences into Festival containing homographs.

Type	POS tagging
Sentences	"It is no use to use the MacBook Pro"
mistakes	See as Table 5.

Table 6. Mistake of POS tagging

Tokens	**It**	**is**	**no**	**use**	**to**	**use**	**the**	**MacBook**	**Pro**
POS tagging	prp	vbz	rb	vb	to	vb	dt	jj	nn
answer	√	√	X	X	√	√	√	X	√

Table 7. POS tagging for an example

As we can see in Table 6 and 7, Festival identifies the first word ‘use’ as a verb rather than noun, ‘no’ as an adverb, and the MacBook as an adjective. I thought it is because it made mistakes at the POS tagging stage. In order to solve this, it may be helpful to create a Homograph dictionary list, and use the word sense disambiguation algorithm like decision-list algorithm.

3.3 Phrase break prediction

When we input the sentence into Festival as Table 8, there should be a break before the word “and” in this case. There is something wrong that the Festival phrase break prediction model cannot effectively predict the break. I think it is necessary to re-label the training set and re-fit the classification tree.

Type	**Phrase break prediction**
Sentences	" They don't like each other— and with good reason. "
mistakes	There should be a break before “and”.

Table 8. Phrase break prediction mistake

Tokens	**They**	**don't**	**like**	**each**	**other**	**—**	**and**	**with**	**good**	**reason**	**.**
POS tagging	NB	NB	NB	NB	NB	NB	NB	NB	NB	NB	
Prediction	√	√	√	√	√	X	√	√	√	√	

Table 9. test utterance for phrase break prediction mistake

3.4 Pronunciation

The pronunciation of word sequence "bass guitar" in Table 10 should be /b ey s/, rather than /b ae s/.

Type	**Pronunciation**
Sentences	"bass guitar"
mistakes	Bass pronunciation should be /b ey s/, rather than /b ae s/

Table 10. Pronunciation mistake

Festival should go through the context for words which may have different meanings for the same tag. In this case, when the word *bass* modifies word *fish*, it should be pronounced by /b ae s/. However, if it modifies word *guitar*, it would be pronounced by /b ey s/. In my opinion, Festival should consider the context, especially the previous and next several words into consideration when dealing with this situation.

3.5 Waveform generation

When Festival generates the word “Beijing”, there is an obvious fault in the final diphone, as we can see as Figure 7 below. This is because the diphone *zh_i* is missing in Festival. We should add more diphones into our pre-recording database in order to avoid this problem.

Type	**Waveform generation**
Sentences	"Hello Beijing"
mistakes	Correct pronunciation analysis but wrong speech

Table 11. Waveform generation mistakes

```
festival> (Wave_Synth myutt)
Missing diphone: zh_i
 diphone still missing, backing off: zh_i
 backed off: zh_i -> zh_@
 diphone still missing, backing off: zh_@
 backed off: zh_i -> #_@
```

Figure 6. Waveform synthesis error

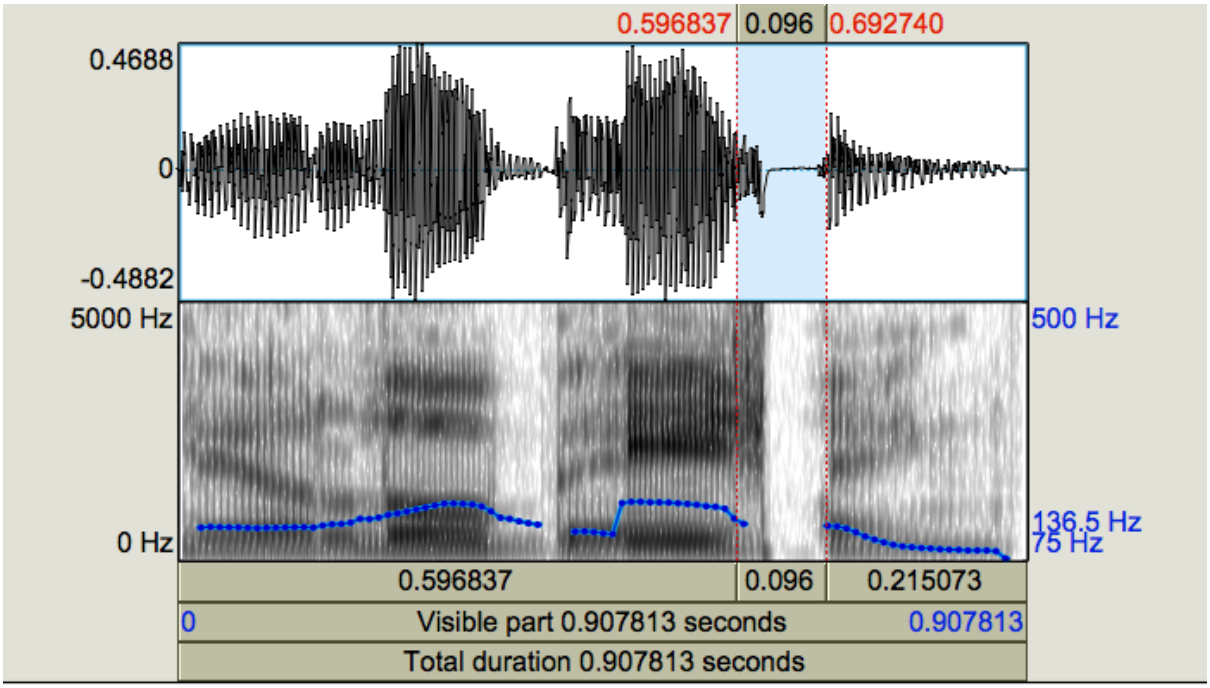

Figure 7. Festival generated Waveform of utterance “Hello Beijing”

3.6 Other types of mistakes

There is no diphone z_uuu for the pre-recording collection of diphone. When Festival generate a diphone of “z_uuu”, it will occur a diphone missing mistake.

Type	**Diphone missing**
Sentences	"Do you live near a zoo with live animals"
mistakes	diphone missing, backing off: z_uuu backed off: z_uuu -> z_@

Table 12. Other types of mistake

In this case, Festival should pre-record more diphones in the database, or fit a larger corpus of training data to predict this diphone.

4. Discussion and conclusion

Although Festival is a powerful toolkit for speech synthesis, there are still many different situations that it did not take into consideration. For example, it is very important to deal with

the word sense disambiguation, which may not only affect the POS tagging results but also the pronunciation results. In order to tackle this, Festival should build better machine learning-based model to make the speech generation precise. Besides, speech synthesis consists of a series of procedures, whichever part's result could directly affect next few steps. There are also many up-to-date machine learning techniques we could employ to take replace of CART decision tree in some complicated situations.

References

Clark, R.A., Richmond, K. and King, S., 2007. Multisyn: Open-domain unit selection for the Festival speech synthesis system. Speech Communication, 49(4), pp.317-330.

Jurafsky, D. and Martin, J.H., 2014. Speech and language processing (Vol. 3). London: Pearson.

King Simon (2017) LASC10065, Speech Synthesis Lecture Pack. University of Edinburgh

Taylor, P., 2009. Text-to-speech synthesis. Cambridge university press.

Part II: Mini literature review

Prosody and intonation precision may affect the auditory sense of audience, which also plays a crucial part in the criteria of speech synthesis evaluation. There are many researches focusing on camparing and evaluating the prosody and intonation model.

At first, many scholars combined decision tree method when building up their prediction model. Syrdal et al. (1998) compared three different methods based on CART tree: rule-based model using ToBI labels, Tilt model and parametric intonation event (PaIntE) model with Vector Quantization (VQ), using the database of Wall Street Journal (WSJ) corpus and Prompt corpus of telephone network services recordings. Authors built six different models to evaluate the performance based on twelve random selected utterances with two different speech synthesizers, HNM and PSOLA. The results showed that the VQ systems have a stable performance whilst the Tilt system has the lowest mean rating among all the test models. In addition, Syrdal et al. (1998) claimed that the bad performance for Prompt data maybe due to the tiny scale of Prompts database and it may be difficult to model Prompt material. However, I hold the view that authors should give a further exploration before making this hypothesis because it is not sufficient to use only one Prompt corpus and only six utterances to support this idea.

Meanwhile, some researches were focusing on the performance between different models. Fordyce and Ostendorf (1998) compared the performance of method of decision tree and transformational rule-based learning (TRBL) in terms of predicting pitch accent location and phrase boundary. They demonstrated that TRBL could have slightly better improvement than decision tree and good phrase structure could help with pitch prediction procedure. Nevertheless, part-of-speech(POS) tagging errors were corrected for test set but not for training set in their experiment. This means they have fitted the model based on training data which may make POS mistakes, and this may lead to pronunciation errors as well as the pitch accent location and phrase boundary errors. In this case, it could be better to control the same research conditions in this experiment.

With the prevalence of neural network, there are more researchers training neural networks for prosody and intonation prediction. Rao (2011) trained four-layer feedforward neural network(FFNN) for multiple-language intonation with selected 25 positional, contextual and phonological features and showed that FFNN could perform a good quality when completing intonation prediction task. FFNN-based applications were also demonstrated to be a satisfying result when dealing with different tasks, such as speech synthesis, speech recognition and language identification.

In conclusion, various machine learning methods have been employed when predicting prosody and intonation. It is important to select representative features, choose satisfying data set and compare different models. Comparing and combining different models could be a good point to improve the performance of prosody and intonation prediction.

References

Fordyce, C.S. and Ostendorf, M., 1998, December. Prosody prediction for speech synthesis using transformational rule-based learning. In ICSLP.

Rao, K.S., 2011. Role of neural network models for developing speech systems. Sadhana, 36(5), pp.783-836.

Syrdal, A., Moehler, G., Dusterhoff, K.E., Conkie, A. and Black, A.W., 1998. Three methods of intonation modeling.