1. 提取参数方法，有两大类：

* 使用world提取(mgc+bap+lf0)
* 使用reaper+straight(lsp+lf0)

1. 文本信息处理有两种：

* 对齐
* 精标按比例

1. 提参、训练方法

首先对于音频的处理，对音频降采样为16KHz，node1

/home/disk5/chenyi/tools/data\_prepare/feature\_extract/down\_sample.sh

/home/disk5/chenyi/tools/data\_prepare/feature\_extract/down\_sample.py

去掉静音段（静音段一般为0.5s）

/home/disk5/chenyi/tools/data\_prepare/feature\_extract/remove\_sil.sh

注：对于两种提参方法，都要进行以上两步。

## 提参：

### World

* 1. 使用world提取声学参数

/home/disk5/chenyi/tools/data\_prepare/feature\_extract/extract\_features\_for\_merlin.sh

* 1. 提取好声学参数，进行copy\_syn，合成语音，查看提取的参数是否有问题

/home/disk5/chenyi/tools/data\_prepare/feature\_extract/extract\_melin/synthesize.sh

* 1. 提取特征后，需要对提取的谱和lf0特征进行处理，并加上UV信息，最后将声学特征进行合并成cmp文件，这些操作均在脚本acoustic\_composition.py中实现，执行该脚本即通过提取好的特征文件生成二进制的ls文件，注意113-119的维度信息

Mgc：60

Bap：5

Lf0：1

Vuv：1

/home/disk5/chenyi/tools/data\_prepare/feature\_extract/extract\_melin/ acoustic\_composition.py

### Reaper+straignt

Node8：/home/yichen/tools/data\_prepare/reaper+straight\_extract

1. get\_f0.sh ，大致设置里面的 maxf0 和 minf0，尽量宽泛

结果得到f0文件。

1. 使用f0\_stats\_pic.py可以得到f0文件夹内f0的上下限，得到max 和min 回带入get\_f0.sh 进行第一步，得到准确的f0，也可使用脚本。。。画出f0图像
2. make（make前修改MAKEFILE信息，设置maxf0和minf0）

得到lf0 和lsp(为了方便（world），起名为mgc)

检验：

1）./vocoder\_bd /getwav\_bd\_vocoder.py 可得到.pcm 和.wav文件。

2）./vocoder\_lsp/gain\_wav.py

检验过程是对提取的声学参数的检测，看提取是否正确。效果不好，重新提惨

（2）提取特征后，需要对提取的谱和lf0特征进行处理，并加上UV信息，最后将声学特征进行合并成cmp文件，这些操作均在脚本acoustic\_composition.py中实现，执行该脚本即通过提取好的特征文件生成二进制的cmp文件

./acoustic\_composition\_lsp.py （附带文件：acoustic\_base.py）

两套提参方法都是用这个文件进行合成cmp，第一个用到lf0、mgc、bap第二套使用lf0、mgc（lsp），因此需要注释掉bap内容，同时维度方面

World:

Lf0:9

Bap:5

Mgc:60

Uv:1

Reaper+straight:

Ls0:9

Uv:1

Lsp:41

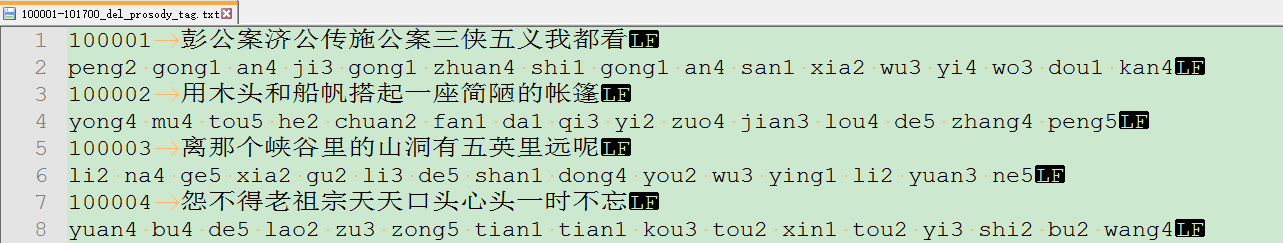
## 提LABEL

Node8:/home/yichen/tools/data\_prepare/gen\_label

1. 文本数据准备，将要数据文本准备成如下格式：
2. 精标数据需要去掉韵律标记（即：#1.#2，#3等）

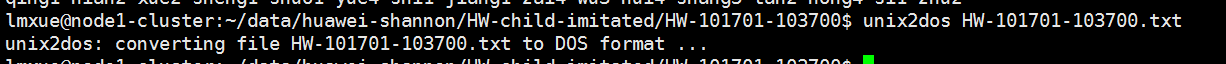
/home/yichen/tools/data\_prepare/gen\_label/del\_prosody\_tag.py

1. 注意：
2. 最后的换行符是LF，而不是CRLF



如果文件出现乱码，则将文件转换成dos格式，命令如下：

unix2dos filename，如下：



或者 iconv命令，将gbk转utf-8，命令如下：

iconv -f gbk –t sourceFile > targetFile

例如：iconv -f gbk -t utf-8 HW-100001-105000.txt > HW-100001-105000\_utf-8.txt

使用线上系统，tts\_system/core/bin/extract\_label.cc 提取label，会生成，文本格式和onehot格式两种。

## 预测数据准备

1. 执行脚本进行状态对齐

/home/disk5/chenyi/tools/data\_prepare/gen\_frame\_label/forced\_alignment.py

1. 执行该脚本需要指定wav的位置、字符串格式label的位置（文本）、训练的句子id列表（没有后缀名）以及最后生成的对齐数据位置

获得文件的名字列表方法：（使用 get\_list.py）

ls HW-100001-105000-txt-label/ |cut –d ‘.’ –f 1 > HW-100001-105000-id-list

该命令可以根据需要进行修改分别可得到只有名字的、有名字和后缀的列表、scp文件、txt文件等。

b) 最后生成状态对齐文件：label\_state\_align（此外，还会生成config

、mfc、model、mono\_no\_align等文件夹）。

1. 从状态对齐文件中提取时间信息，脚本文件：

/home/disk1/xiaochunan/workspace/nnet/data\_prepare/gen\_frame\_label/prepare\_dur\_out.py

执行该脚本需要时：

* 需要指定状态对齐文件的路径，以及生成状态时间信息的文件夹路径（脚本第82~84行）
* state\_info的值（第87行），当state\_info = True时，是状态对齐；当state\_info=False时，是音素对齐。这里仅需要对状态对齐后的时间信息提取出来。
* vim li 生成dur\_out文件夹

1. 生成神经网络需要的label

(1)得到上述包含时间信息的label后，需要将其转化为one-hot形式，执行tts\_system/bin目录下的test-genacc-in就可以得到神经网络的输入——帧级程度数值label。 注意维度！！！

执行命令：

test-genacc-in list dur\_in dur\_out acc\_in。

* list是文件的索引目录列表（文件id+后缀名），
* dur\_in是label（二进制）矩阵文件夹路径
* dur\_out是方才提取的时间信息文件夹路径
* acc\_in是程序输出的帧级label文件夹路径。

1. 准备prepared\_label 和prepared\_cmp

执行脚本：

/home/disk2/chenyi/tools/data\_prepare/gen\_frame\_label/make\_data.py

（在要生成prepared\_label/prepared\_cmp 的路径下执行该脚本

a) 脚本作用：对得到的cmp 进行一定的谱平滑，然后根据帧级数值label和cmp 的长度，把多余的帧截断，并对f0加上下文信息。之后将输入输出以二进制格式分别写入prepared\_label 和prepared\_cmp 文件夹中。make\_data.py脚本中的 f0\_context = 4，是对f0添加了上下文信息，即插值，先后分别插入4维，最后的lf0共9维。

b)数据准备： 脚本内需要train.scp文件，里面是文件索引目录（不需要后缀名）。

c) 修改cmp和acc\_in两个文件夹的路径(第53行和55行)

d) 最后会根据acc\_in和cmp两个文件夹生成最后需要的数据prepared\_label和prepared\_cmp，需要修改prepared\_label和prepared\_cmp的路径

e) 执行make\_data.py时，报错：

解决方法：pip install scipy

1. 模型训练（TensorFlow）

## 精标按比例数据准备

Node1：/home/yichen/tools/data\_prepare/gen\_label

中文精标按比例步骤

1. 生成精标的label（包括带时长和不带时长）
2. 使用label（不带时长）做对齐，得到dur\_out1（每个音素的状态分别有多少帧），五状态的话每个音素每行五列，分别持续多少帧。
   1. 执行脚本进行状态对齐

/home/yichen/tools/data\_prepare/gen\_frame\_label/forced\_alignment.py

1. 执行该脚本需要指定wav的位置、字符串格式label的位置（文本不带时长）、训练的句子id列表（没有后缀名）以及最后生成的对齐数据位置

获得文件的名字列表方法：（使用 get\_list.py）

ls HW-100001-105000-txt-label/ |cut –d ‘.’ –f 1 > HW-100001-105000-id-list

该命令可以根据需要进行修改分别可得到只有名字的、有名字和后缀的列表、scp文件、txt文件等。

1. 最后生成状态对齐文件：label\_state\_align（此外，还会生成config

、mfc、model、mono\_no\_align等文件夹）。

* 1. 从状态对齐文件中提取时间信息，脚本文件：

/home/yichen/tools/data\_prepare/gen\_frame\_label/prepare\_dur\_out.py

执行该脚本需要时：

* 需要指定状态对齐文件的路径，以及生成状态时间信息的文件夹路径（脚本第82~84行）
* state\_info的值（第87行），当state\_info = True时，是状态对齐；当state\_info=False时，是音素对齐。这里仅需要对状态对齐后的时间信息提取出来。
* 生成dur\_out1文件夹

1. 使用prepare\_dur\_out.py脚本处理上一步前端系统得到的带时长的label，将state\_info参数设置为false,得到每个音素的持续时长dur\_out2,只有1列。
2. 使用scale.py脚本进行精标按比例分配，force\_dir =dur\_out1，精标提取的每个音素持续帧数；exact\_dir=dur\_out2，对齐得到的每个状态持续帧数，由于dur\_out1每个音素所有状态持续的总帧数并不一定等于dur\_out2精标音素持续帧数，所有根据dur\_out1中每个状态所占的比例，对dur\_out2精标时长进行帧数的重新分配。参数3=dur\_out为输出时长的路径。
3. 需要注意的是对每个状态数求比例的时候，每个状态至少持续一帧，后续步骤均与平常相同，得到dur\_out，获得acc-in.

得到上述包含时间信息的label后，需要将其转化为one-hot形式，

执行get\_acc\_in.sh

注意维度！！！

执行命令：

test-genacc-in list dur\_in dur\_out acc\_in。

list是文件的索引目录列表（文件id+后缀名），

dur\_in是label（二进制）矩阵文件夹路径

dur\_out是方才提取的时间信息文件夹路径

acc\_in是程序输出的帧级label文件夹路径

1. 准备prepared\_label 和prepared\_cmp

执行脚本

/home /chenyi/tools/data\_prepare/gen\_frame\_label/make\_data.py

a) 脚本作用：对得到的cmp 进行一定的谱平滑，然后根据帧级数值label和cmp 的长度，把多余的帧截断，并对f0加上下文信息。之后将输入输出以二进制格式分别写入prepared\_label 和prepared\_cmp 文件夹中。make\_data.py脚本中的 f0\_context = 4，是对f0添加了上下文信息，即插值，先后分别插入4维，最后的lf0共9维。

b)数据准备： 脚本内需要train.scp文件，里面是文件索引目录（不需要后缀名）。

c) 修改cmp和acc\_in两个文件夹的路径(第53行和55行)

d) 最后会根据acxc\_in和cmp两个文件夹生成最后需要的数据prepared\_label和prepared\_cmp，需要修改prepared\_label和prepared\_cmp的路径

e) 执行make\_data.py时，报错：

解决方法：pip install scipy

1. 模型训练（TensorFlow）

## 加mlpg

1、与不加mlpg不同之处是，在生成cmp时是生成的一阶二阶差分的mlpg

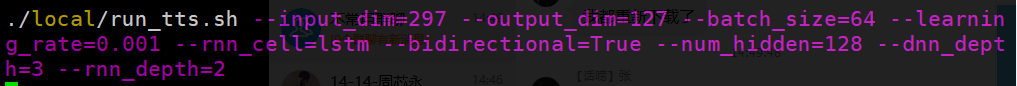
使用node1

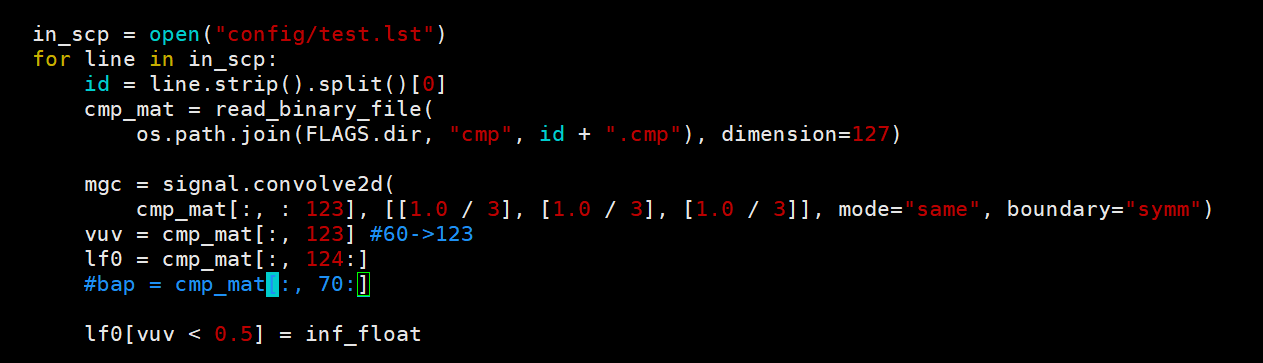
/home/disk5/chenyi/tools/data\_prepare/reaper\_straight\_lsp/ acoustic\_composition\_lsp.py

重新合成一阶二阶差分的cmp，

2、然后使用该目录的make\_data\_lelat.py，然后训练网络，

3、训练时，更改维度，





4、训练完将测试集的cmp拷贝到node1：/home/disk5/chenyi/workspace/mlpg-extract

使用get\_cmvn\_convert\_dat.py算出全局方差，生成var文件

使用split\_cmp\_lsp.py将vuv信息单独写到文件

然后使用parameter\_generation.py做mlpg

对lf0进行修正：使用 repair\_lf0.py

对得到的f0和mgc 使用bd\_vocoder合成。

# 模型训练

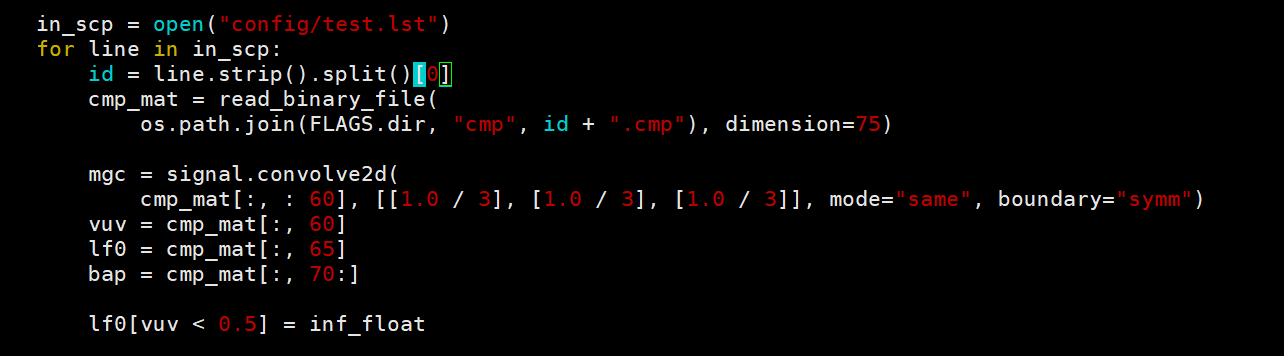
## 声学模型：

以上部分准备的数据位声学模型的数据

1. 参照张雨超文档《神经网络语音合成工具包tf-tts说明文档.pdf》中的模型训练内容：



Split\_cmp.py



1. 运行run\_lstm\_tts.sh时，可以添加长度的参数，可以参考：

/tf-tts-master/src/run\_lstm\_tts.py

1. 运行时设置学习率，调参主要就是指调整学习率参数。
2. 关于维度：如果是tts system就是上图所述维度

## 时长模型

### 数据

在声学模型的数据准备中，我们在test-genacc-in这一程序中使用的dur\_in(one-hot格式的label) dur\_out(对齐得到的时长信息) 的这两个文件夹即为时长模型的输入和输出数据。

需要扩帧处理，上下两帧，脚本如下，Node8

/home/disk1/xyzhou/data/roobo\_boy\_mix/extern\_dur\_in.py

输入dur\_in，输出扩帧后的dur\_in

### 特征准备

运行feature\_prepare\_duration.m脚本生成prepared\_cmp和prepare\_label文件夹，接下里的准备步骤和声学模型数据准备步骤一致。其中feature\_prepare\_duration.m脚本用到了readBin.m和writeBin.m这两个脚本，需放在同一文件目录下。输入为扩帧后的dur\_in和dur\_out

Node8: /home/disk1/xyzhou/scripts/dur\_model\_script

### 模型训练

修改run\_tts.sh。参考node8:

/home/disk1/xyzhou/workspace/tf\_workspace/ch\_en\_mix/voicenet/egs/dur\_model\_roobo\_mix/local/run\_tts.sh

两者不同主要是模型结构不一样，其余的参数设置和声学模型参数设置方法一样。

参考node8:

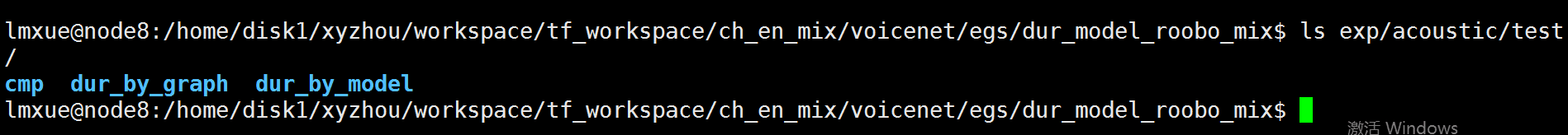
/home/disk1/xyzhou/workspace/tf\_workspace/ch\_en\_mix/voicenet/src/ run\_tts\_dnn.py

/home/disk1/xyzhou/workspace/tf\_workspace/ch\_en\_mix/voicenet/src/ models/tf\_model\_dnn.py

计算误差脚本在node8

/home/disk1/xyzhou/workspace/tf\_workspace/ch\_en\_mix/voicenet/misc/scripts/dur\_rmse.py

测试时长模型

训时长模型，用freeze\_graph.py导出图之后，执行stage=5的src/decode\_by\_graph.py  
然后exp/acoustic/test/ 下会生成dur\_by\_graph这个就是导出图预测的结果;

导出图就是把checkpoint保存的参数填充到计算图中