# Project 2：Kaldi 的安装和基本使用

*520030910155* 邱一航

## 0. 安装环境

笔者在 Windows 提供的 Linux 子系统，即 WSL（Windows Subsystem for Linux）环境下安装 Kaldi。

所安装的 WSL 版本为 2.0，笔者在该子系统中安装 Ubuntu 操作系统，Ubuntu 版本为 Ubuntu 20.04。WSL 环境下已预先安装了 CUDA 10.1。

## 1. Kaldi 的安装

从 https://github.com/kaldi-asr/kaldi 下载 Kaldi 源码，解压后根据 INSTALL 文件的提示逐步操作即可完成 Kaldi 源码的编译和相关安装。

### 1.1. Kaldi 源码编译过程

#### *1.1.1.* `tools` 文件夹内的编译

首先运行 `extras/check_dependencies.sh` 文件，该文件会检查 Kaldi 所需要的前置条件是否满足。根据指示安装 automake、autoconf 等 Kaldi 所需要的库，并安装 Intel MKL。

之后运行 `make -j 4`，使用四个 CPU 核来完成 Kaldi 所需库的编译。

由 `src` 内 INSTALL 文件可知，这一阶段完成了 OpenFst 的编译与 ATLAS 库、CLAPACK 库的头文件获取。

#### *1.1.2.* `src` 文件夹内的编译

运行 `./configure –shared`，为 Kaldi 的编译即可完成 MKL 的配置。

之后运行 `make depend -j 4` 和 `make -j 4` 完成 Kaldi 源码的编译。

### 1.2. 遇到的困难

#### *1.2.1.* 包管理器软件包列表更新

根据 `tools/extras/check_dependencies.sh` 的运行结果进行 Kaldi 所需库的安装时，安装失败并返回以下错误信息。

```
E: Unable to fetch some archives, maybe run apt-get update or try with --fix-missing?
```

**解决**：发现是 Ubuntu 包管理器软件包列表并非最新版导致的。根据提示运行 `apt-get update` 后重新进行 Kaldi 所需库的安装即可。

#### *1.2.2. WSL* 管理员权限

安装 MKL 时提示需要管理员权限。

**解决：**使用以下指令运行 `install_mkl.sh`。

```
sudo extras/install_mkl.sh -sp debian intel-mkl-64bit-2020.0-088
```

#### *1.2.3. MKL* 配置

在 `src` 目录下运行 `./configure –shared` 进行 MKL 配置时，提示错误如下。

```
***configure failed: CUDA 10_1 does not support c++ (g++-9).
You need g++ < 9.0. ***
```

**解决**：查看 WSL 环境下 g++ 编译器版本，发现是 9.4.0，而 CUDA 10.1 不支持该版本的 g++ 编译器。笔者尝试下载 g++-4.8 版本，发现无论使用 apt、aptitude、apt-get 都返回如下错误。

```
E: Package 'g++-4.8' has no installation candidate
```

大概率是由于 Ubuntu 20.04 的软件包列表中已经不支持较低版本的 g++ 和 gcc 编译器。因此笔者转而下载 g++-7（7.5.0 版本）并将默认编译器切换为 g++-7。再次使用 `./configure –shared` 进行 MKL 配置，配置成功。

### 1.3. Kaldi 的基本结构

Kaldi 编译后的源码的根目录共有 8 个文件夹，它们的内容分别如下。

| 目录名 | 内容 |
|---|---|
| cmake | 使用 C++ 进行源码编译的文件。 |
| egs | 一些 Kaldi 使用的示例代码。 |
| misc | Kaldi 相关的论文及 htk 脚本。 |
| scripts | 包含了 RNNLM 的实现和 Wakeword。 |
| src | 包含了 Kaldi 的源码。 |
| | 包含了 Kaldi 依赖库的安装脚本。 |
| tools | 包括 OpenFst、ATLAS、CLAPACK、IRSTLM、SRILM 等。 |
| windows | 在 Windows 系统下安装所使用的脚本。 |

## 2. Kaldi 的基本使用

笔者使用了 egs 目录下的 yesno 这一示例来尝试使用 Kaldi。

### 2.1. 示例程序运行中的困难

直接在对应目录下运行 ./run.sh，过程中显示了两类错误。

第一类错误是在读取环境变量时发生错误。这里主要是路径中含有空格的问题，而 WSL 的环境变量默认与 Window 系统环境变量挂钩，因存在含有空格的路径（如 C:/Program Files/）。因此读取环境变量 PATH 时出错（在空格处自动截断了，读到了“Files”开始的一串字符串）。

```
sh: 1: export: Files/WindowsApps/
    CanonicalGroupLimited.Ubuntu20.04
    onWindows_2004.2022.8.0_x64__79rhkp1fndgsc:/
    mnt/e/Program: bad variable name
--> ERROR: data/lang/L.fst is not olabel sorted
```

**解决**：使用以下指令修改 WSL 环境下的环境变量 $PATH。（第一行是查看 $PATH 的内容，第二行是修改 $PATH。

```
echo \$PATH
export PATH=home/balthasar/.local/bin:/usr/local/
    sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin
    :/bin:/usr/games:/usr/local/games:/usr/lib/
    wsl/
```

第二类错误是“未找到对应文件”，错误信息如下。

```
local/prepare_data.sh: line 39: utils/
    utt2spk_to_spk2utt.pl: No such file or
    directory
```

检查发现 utils 文件指向另一个目录下的 utils 文件夹，不知道为什么 WSL 未能成功实现 utils 的重定向。

**解决**：笔者暂时将这类文件都用所指向的文件或文件夹替代，将目标文件和文件夹复制到当前目录下。这样处理后程序确实能正常运行。

两类错误应该都和 WSL 的系统环境变量 $PATH 相关。但笔者暂未找到更好的解决方案。

### 2.2. 示例程序的基本结构

由 run.sh 可知，yesno 示例的训练过程如下：

第一阶段是数据下载、预处理与模型准备。从指定源下载 waves_yesno 数据集，随后使用 prepare_data.sh 和 prepare_dict.sh 准备数据和发音字典，用 utils/prepare_lang.sh 和 local/prepare_lm.sh 准备状态机。

第二阶段是特征提取。从准备好的数据中提取声学特征，作为后续声学模型的输入。该示例中提取了 MFCC 特征，即梅尔滤波器倒谱系数。

第三阶段是声学模型训练。本示例中训练使用单音素建模的状态机。

第四阶段是解码与效果测试。在测试集上测试训练后的模型的效果，并计算整个测试集上的词错误率（WER）。

### 2.3. 运行结果展示

```
add-self-loops --self-loop-scale=0.1 --reorder=true exp/mono0a/final.mdl exp/mono0a/graph_tgpr/HCLGa.fst
steps/decode.sh --nj 1 --cmd utils/run.pl exp/mono0a/graph_tgpr data/test_yesno exp/mono0a/decode_test_yesno
decode.sh: feature type is delta
steps/diagnostic/analyze_lats.sh --cmd utils/run.pl exp/mono0a/graph_tgpr exp/mono0a/decode_test_yesno
local/score.sh --cmd utils/run.pl data/test_yesno exp/mono0a/graph_tgpr exp/mono0a/decode_test_yesno
local/score.sh: scoring with word insertion penalty=0.0,0.5,1.0
%WER 0.00 [ 0 / 232, 0 ins, 0 del, 0 sub ] exp/mono0a/decode_test_yesno/wer_10_0.0
balthasar@MAGI-Casper3:/mnt/e/_Programs/kaldi/egs/yesno/s5$
```

图 1: *The Result of* run.sh

运行 run.sh 后结果如上图。结果显示 WER 为 0%，在测试集上准确率很高。