# Windows下安装kaldi

安装环境：

Win10+VS2017（Win7经测试也可以）VS2015大概也能编译

预装工具：git，cmake，以下$开头的命令都是在git bash中运行

步骤：

1. 下载kaldi（如果git失败，可以直接去https://github.com/kaldi-asr/kaldi下载）

$ git clone https://github.com/kaldi-asr/kaldi.git kaldi --origin upstream

$ cd kaldi

1. 安装openfst（参考$(kaldi)/windows目录下的INSTALL.md文件）

2.1 下载openfst

$ git clone https://github.com/kkm000/openfst.git

$ cd openfst

$ mkdir build64

$ cd build64

2.2 将（openfst）/src/include/fst/test目录下的所有.h文件（fst\_test.h weight-tester.h algo\_test.h）复制到(openfst)/src/test（？帮我确认一下），然后执行命令：

$ cmake -G "Visual Studio 15 2017 Win64" ../

运行结果：

-- The C compiler identification is MSVC 19.11.25547.0

-- The CXX compiler identification is MSVC 19.11.25547.0

-- Check for working C compiler: C:/Program Files (x86)/Microsoft Visual Studio/2017/Community/VC/Tools/MSVC/14.11.25503/bin/Hostx86/x64/cl.exe

-- Check for working C compiler: C:/Program Files (x86)/Microsoft Visual Studio/2017/Community/VC/Tools/MSVC/14.11.25503/bin/Hostx86/x64/cl.exe -- works

-- Detecting C compiler ABI info

-- Detecting C compiler ABI info - done

-- Check for working CXX compiler: C:/Program Files (x86)/Microsoft Visual Studio/2017/Community/VC/Tools/MSVC/14.11.25503/bin/Hostx86/x64/cl.exe

-- Check for working CXX compiler: C:/Program Files (x86)/Microsoft Visual Studio/2017/Community/VC/Tools/MSVC/14.11.25503/bin/Hostx86/x64/cl.exe -- works

-- Detecting CXX compiler ABI info

-- Detecting CXX compiler ABI info - done

-- Detecting CXX compile features

-- Detecting CXX compile features - done

-- The following ICU libraries were not found:

-- data (required)

-- i18n (required)

-- io (required)

-- test (required)

-- tu (required)

-- uc (required)

-- Failed to find all ICU components (missing: ICU\_INCLUDE\_DIR ICU\_LIBRARY \_ICU\_REQUIRED\_LIBS\_FOUND)

-- Could NOT find ZLIB (missing: ZLIB\_LIBRARY ZLIB\_INCLUDE\_DIR)

-- Configuring done

-- Generating done

-- Build files have been written to: C:/Users/jtrmal/Documents/openfst/build64

3. 编译kaldi。数学计算库可以选择MKL，也可以选择OpenBLAS，本说明使用OpenBLAS，因为MKL是商用的而且需要注册，OpenBLAS是开源的。

3.1 进入kaldi的tools目录下，下载OpenBLAS。

(kaldi)/tools$ curl -L -O http://sourceforge.net/projects/openblas/files/v0.2.14/OpenBLAS-v0.2.14-Win64-int32.zip

(kaldi)/tools$ curl -L -O http://sourceforge.net/projects/openblas/files/v0.2.14/mingw64\_dll.zip

(kaldi)/tools$ unzip OpenBLAS-v0.2.14-Win64-int32.zip

(kaldi)/tools$ unzip mingw64\_dll.zip

3.2 进入kaldi的windows目录，获取windows目录所在路径：

(kaldi)/$ cd windows

(kaldi)/windows $ pwd

3.3 在windows目录下，将文件variables.props.dev名字改为variables.props，用pwd获取的路径替换文件variables.props中相应的内容。

3.4 在windows目录下，将文件kaldiwin\_openblas.props名字改为kaldiwin.props

3.5 执行以下命令生成MSVC解决方案[[1]](#footnote-1)

$ generate\_solution.pl --vsver vs2017 --enable-openblas

如果不能识别generate\_solution.pl，在前面加上perl，即

$ perl generate\_solution.pl --vsver vs2017 --enable-openblas

3.6 执行以下命令：

(kaldi)/windows$ perl get\_version.pl

3.7 kaldi目录下会生成一个子目录，目录名格式：(kaldi)/kaldiwin\_vs<version>\_<blas-library>，例如kaldiwin\_vs2017\_OPENBLAS，进入目录kaldiwin\_vs2017\_OPENBLAS打开kaldiwin\_vs2017.sln，编译该MSVC解决方案。

4. PortAudio的更新及编译。步骤3.7中编译会遇到找不到portaudio.h的错误，按照以下步骤解决该错误（参考文章https://blog.csdn.net/qq\_25867649/article/details/78356474）：

4.1 下载 pa\_stable\_v190600\_20161030.tgz（http://www.portaudio.com/archives/pa\_stable\_v190600\_20161030.tgz）和asiosdk.zip（http://www.steinberg.net/sdk\_downloads/asiosdk2.3.zip），复制到 (kaldi)/tools/目录。

4.2 执行以下命令：

$ cd (kaldi)/tools/

$ tar -xvf pa\_stable\_v190600\_20161030.tgz

$ unzip asiosdk.zip

$ mv ASIOSDK2.3/ portaudio/src/hostapi/asio/ASIOSDK

4.3 打开(kaldi)\tools\portaudio\build\msvc\portaudio.sln，选择x64/Release配置进行编译。

4.4 打开(kaldi)\kaldiwin\_vs2017\_OPENBLAS\kaldiwin\variables.props, 添加PortAudio相关信息：

在<PropertyGroup Label="UserMacros">子项下添加:

<PORTAUDIO>D:\Local\\_Work\Proj\_DL\kaldi\_win\_valid\tools\portaudio</PORTAUDIO>

<PORTAUDIOLIB>D:\Local\\_Work\Proj\_DL\kaldi\_win\_valid\tools\portaudio\build\msvc\x64</PORTAUDIOLIB>

在<ItemGroup>子项下添加:

<BuildMacro Include="PORTAUDIO">  
<Value>$(PORTAUDIO)</Value>  
<EnvironmentVariable>true</EnvironmentVariable>  
</BuildMacro>  
<BuildMacro Include="PORTAUDIOLIB">  
<Value>$(PORTAUDIOLIB)</Value>  
<EnvironmentVariable>true</EnvironmentVariable>  
</BuildMacro>

4.5 打开(kaldi)\kaldiwin\_vs2015\_OPENBLAS\kaldiwin\kaldiwin.props, 添加PortAudio相关信息：

在<AdditionalIncludeDirectories>子项下添加:  
$(PORTAUDIO)\include;$(PORTAUDIO)\src\common;  
  
在<AdditionalLibraryDirectories>子项下添加:  
$(PORTAUDIOLIB)\Debug;  
  
在<AdditionalDependencies>子项下添加:  
portaudio\_x64.lib;pa\_ringbuffer.obj;

4.6 重新编译(kaldi)/ kaldiwin\_vs2017\_OPENBLAS目录下的kaldiwin\_vs2017.sln 的 oneline-wave-gmm-decode-faster工程，如果编译通过就是成功了。

5. 编译成功后，如果要成功运行测试例子，还需要添加一些dll文件到(kaldi)\kaldiwin\_vs2017\_OPENBLAS\x64\Debug目录下，所需dll及所在目录如下：

libopenblas.dll

（KALDI\_ROOT)\tools\OpenBLAS-v0.2.14-Win64-int32\bin\libopenblas.dll

portaudio\_x64.dll

（KALDI\_ROOT)\tools\portaudio\build\msvc\x64\Debug\portaudio\_x64.dll

libgcc\_s\_seh-1.dll

（KALDI\_ROOT)\tools\mingw64\_dll\libgcc\_s\_seh-1.dll

libgfortran-3.dll

（KALDI\_ROOT)\tools\mingw64\_dll\libgfortran-3.dll

libquadmath-0.dll

（KALDI\_ROOT)\tools\mingw64\_dll\libquadmath-0.dll

将以上dll文件复制过去即可。

附1：如何生成cuda版本的kaldiwin-2017.sln

1. 安装CUDA和CuDNN。本例子安装的是CUDA10.1和CuDNN7.5。
2. 将C:\Program Files (x86)\MSBuild\Microsoft.Cpp\v4.0\BuildCustomizations下的四个文件复制到C:\Program Files (x86)\Microsoft Visual Studio\2017\Enterprise\Common7\IDE\VC\VCTargets\BuildCustomizations。（也许CUDA安装的时候会自动把文件复制好了，需要大家确认一下）
3. 打开(kaldi)/windows/variables.props，修改<PropertyGroup Label="UserMacros">下<CUBDIR>的内容，把路径换成本地的CUDA路径，例如：

C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v10.1

确保该路径下有include和lib两个子目录。

1. 打开(kaldi)/windows/cuda\_7.0.props，修改<AdditionalDependencies>内容，加上以下lib文件：curand.lib;cusparse.lib;cusolver.lib。
2. 打开(kaldi)/windows/ generate\_solution.pl，将所有“CUDA 7.0.props”改成“CUDA 10.1.props”，以及“CUDA 7.0.targets”改成“CUDA 10.1.targets”（因为本例子安装的是CUDA 10.1，所以改成10.1）。注意：不要改“cuda\_7.0.props”，这是第4步的文件名。
3. 执行以下命令生成MSVC解决方案

$ perl generate\_solution.pl --vsver vs2017 --enable-openblas --enable-cuda

$ (kaldi)/windows$ perl get\_version.pl

1. 在kaldi目录下会生成kaldiwin\_vs2017\_OPENBLAS\_CUDA目录，
2. 打开(kaldi)/kaldiwin\_vs2017\_OPENBLAS\_CUDA /kaldiwin\_vs2017.sln，编译。

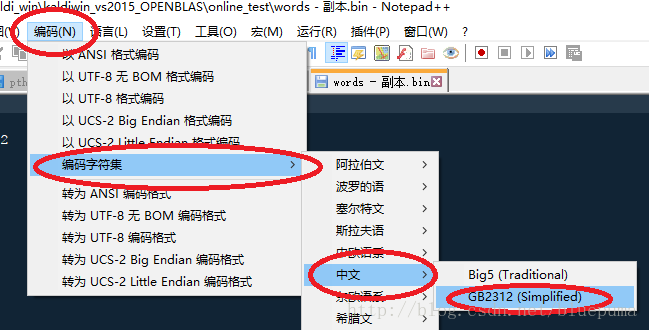
编译可能碰到的错误：

1. 有个Log(\*)函数，提示无法将参数1从<Real>类型转换成fst::\*\*\*<Real>类型，我在Log()前加上kaldi::，变成kaldi::Log(\*\*)，至少编译通过了。
2. 编译工程online-server-gmm-decode-faster时：std::numeric\_limits<int32>::max() 应输入标识符。需要把max用括号括起来避免和windows定义的宏混淆 **(std::numeric\_limits<int32>::max)()** 因为Windef.h中定义了

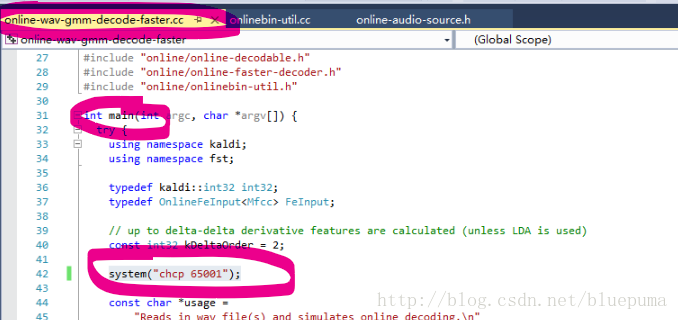
#ifndef max  
#define max(a,b)            (((a) > (b)) ? (a) : (b))  
#endif

所以会产生编译错误

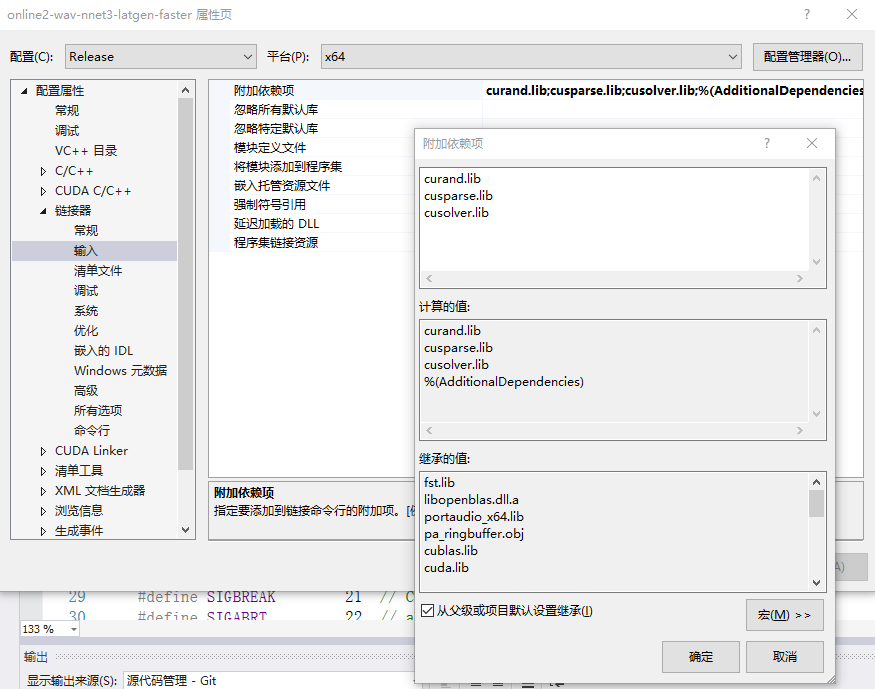
1. 控制台输出的解码后的文本显示是乱码。解码所需的字典文件word.txt的编码格式（UTF-8）和中文版windows的控制台默认的编码格式（GBK）不一致
2. 用notepad++将word.txt编码格式更改为GB2312(GBK)



1. 将控制台输出的编码格式更改为UTF-8（optional好像不改也可以）



1. 生成支持CUDA的kaldi。
   1. 首先对VS设置CUDA路径。将C:\Program Files (x86)\MSBuild\Microsoft.Cpp\v4.0\BuildCustomizations下的四个文件复制到C:\Program Files (x86)\Microsoft Visual Studio\2017\Enterprise\Common7\IDE\VC\VCTargets\BuildCustomizations。修改（kaldi）/windows下generate\_solution.pl，把CUDA 7.0.props和CUDA 7.0.targets改成CUDA 8.0.props和CUDA 8.0.targets（因为本地机器装的CUDA 8.0，如果安装的是9.0就换成相应的版本）。
   2. 修改(kaldi)/windows下variables.props里<CUBDIR>内容，换成本地机器的CUDA路径，例如C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v8.0
   3. 然后就可以在VS2017打开(kaldi)\kaldiwin\_vs2017\_OPENBLAS\_CUDA\kaldiwin\_vs2017.sln
2. 编译到link的时候可能找不到cuda的动态链接库，只要按照错误提示到cuda目录下找到缺的动态链接库的名字加到“附加依赖项”里就行了（见下图）



1. generate\_solution.pl还可以接受其他参数，参数模板如下：

   generate\_solution.pl --vsver <default|vs2017|vs2015> [--enable-cuda] [--enable-openblas] [--enable-mkl]

   具体参考INSTALL.md说明 [↑](#footnote-ref-1)