Windows下在python 2.7中安装fasttext

1. 环境信息

OS：windows7 64bit

Python：2.7.14 [MSC v.1500 64 bit (AMD64)]

2. 安装过程

2.1 安装

1、安装方法有很多，可以直接

pip install fasttext

也可以先clone源码

git clone https://github.com/facebookresearch/fastText.git

2、然后

cd ./fastText

pip install .

或者

python setup.py install

2.2 问题

但是在本文最开始提到的环境中直接安装是**不可能**成功的

2.2.1 缺少支持python2.7的c++编译器

首先你可能会得到一个错误信息

“error: microsoft visual c++ 9.0 is required”

由于python2.7默认的c++编译器版本是9.0，如果没有此版本的编译器或者开发包会报出这个错误，解决方法很简单，以下两种方法取其一即可：

1、 访问http //aka.ms/vcpython27，下载安装Microsoft提供的编译器

2、 下载visual studio 2008并安装

但是，安装完了之后依然安装会**失败。**

2.2.2 “to_string”is not class number of std

这是fasttext源码的一个bug，需要在fasttext/src/productquantizer.cc，中加上#include <string>。

如果这时候报错LNK1104: can not open file 'pthread.lib' error. 解决办法是从setup.py L62中的extra_compile_args去掉-pthread flag

2.2.3 缺少"atomic"头文件

仔细查看报错信息，你会发现，报错提示缺少"atomic"头文件。

但这不是简单的缺失头文件的问题，及时你把这个头文件加进去依旧会有报错。

熟悉C++的同学应该知道，这个头文件引入了原子操作，这是C++11的新特性，然而MS

9.0/VS2008是不支持C++11特性的。

但是fasttext的官方文档中明确说明了其需要C++11的支持。

于是在windows下对于python 2.7安装fasttext就造成了一个矛盾：

Python 2.7使用的是MS 9.0编译器，不支持C++11，然而，fasttext需要C++11的支持

那在windows下，在python 2.7中安装fasttext就不可能了么？**并不是。**

2.3 解决方案

先直接给出解决方案，有兴趣的读者可以继续往下看原因和细节

1、到python安装的地址./Lib/distutils/目录下打开msvc9compiler.py文件

2、找到get_build_version()函数，直接让该函数return 14.0。（当然也可以更高，具体数值参考下文中的介绍，当当然，你的机器中得安装有相应的编译器才行。）

3、回到2.1安装fasttext

发现

```
Successfully built fasttext
Installing collected packages: fasttext
Successfully installed fasttext-0.8.22
```

大功告成

3. 问题解析

整个安装过程中所有的问题归结为一点就是：C++编译器版本问题

由于python 2.7默认的C++编译器是9.0，然而fasttext需要C++11特性，该版本的编译器不支持，所以需要14及以上的编译器版本。

3.1 背景知识

要想理解为什么，需要补充一点背景知识。

3.1.1 编译器的版本

目前，我们使用的Python大部分属于CPython，也就是用C语言实现的Python版本。本质上来讲，无论是在Linux还是Windows平台下，我们使用的都是C编译器编译后的（python）可执行程序。不同点在于，Linux平台源码安装（默认的C编译器一般为gcc）python，C/C++编译环境在本地;Windows平台则通过下载编译好的Python安装使用,本地不一定具备C/C++编译环境。

编译Python使用的编译器版本信息可在python版本信息中查看：

```
Python 2.7.14 |Anaconda custom (64-bit)| (default, Oct 15 2017, 03:34:40) [MSC v.1500 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import sys
>>> sys.version
'2.7.14 |Anaconda custom (64-bit)| (default, Oct 15 2017, 03:34:40) [MSC v.1500 64 bit (AMD64)]'
```

为保证兼容性，使用pip给python添加扩展/模块时，如果扩展/模块中包含有C/C++源码

，安装脚本将试图寻找与编译生成Python的同版本编译器来编译生成该模块。也就是说在上述两个平台下，Linux平台将寻找并使用GCC 5.4.0 来编译生成Python扩展/模块，Windows平台将寻找并使用Visual C++ 9.0 来编译生成Python扩展/模块。

上面说过，编译生成Python的编译器及版本信息可以在Python的版本信息中查看。通常情况下，Linux平台能够保证Python编译生成环境和Python扩展/模块编译生成环境的一致性，使用中不会存在什么问题。所以本节将主要讨论Windows下的C/C++编译器。

Windows平台下的C/C++编译器默认是 Microsoft Visual C++ （下面简称VC）。VC常作为Microsoft Visual Studio（以下简称VS）开发套件的组成部分在 VS安装时被安装在设备上，我们通常不单独安装VC。特别需要注意的是不同版本 的VS搭载不同版本的VC，其对应关系如下：

具体的对应关系

VS版本	内部版本	VC版本
Visual Studio	4.0	Visual C++ 4.0
Visual Studio 97	5.0	Visual C++ 5.0
Visual Studio 6.0	6.0	Visual C++ 6.0
Visual Studio .NET 2002	7.0	Visual C++ 2002
Visual Studio .NET 2003	7.1	Visual C++ 2003
Visual Studio 2005	8.0	Visual C++ 2005
Visual Studio 2008	9.0	Visual C++ 2008
Visual Studio 2010	10.0	Visual C++ 2010
Visual Studio 2012	11.0	Visual C++ 2012
Visual Studio 2013	12.0	Visual C++ 2013
Visual Studio 2015	14.0	Visual C++ 2015
Visual Studio 2015 RTM	14.0	Visual C++ 2015

MS 12.0也不能保证完全支持C++11，建议14.0以上版本

3.1.2 Python 获得VC的版本

上面也提到了，python获得VC编译器版本的代码在./Lib/distutils/msvc9compiler.py文件中，代码如下：

```
def get_build_version():
    """Return the version of MSVC that was used to build Python.
    For Python 2.3 and up, the version number is included in
    sys.version.  For earlier versions, assume the compiler is MSVC 6.
    """
```

```
prefix = "MSC v."
i = sys.version.find(prefix)
if i == -1:
    return 6
i = i + len(prefix)
s, rest = sys.version[i:].split(" ", 1)
majorVersion = int(s[:-2]) - 6
minorVersion = int(s[2:3]) / 10.0
# I don't think paths are affected by minor version in version 6
if majorVersion == 6:
    minorVersion = 0
if majorVersion >= 6:
    return majorVersion + minorVersion
# else we don't know what version of the compiler this is
return None
```

代码比较简单，就不多做解释了。python知道对应的VC版本后，接下了就是找到该版本VC 编译器的物理路径。脚本在寻找VC路径的时候遵循“先注册表，后环境变量”的顺序。对于VC版本>=9.0的情形，脚本通过 HKLM\SOFTWARE\WOW6432Node\Microsoft\VisualStudio\VC版本\Setup\VC\ProductDir (64位Python) 或 HKLM\SOFTWARE\Microsoft\VisualStudio\VC版本\ Setup\VC\ProductDir （其他情况）键值获取VC编译器的路径。若该键不存在则通过 环境变量“VS@0COMNTOOLS”来确定VC 编译器路径，其中@ 为VC主版本号。具体的寻径策略见 msvc9compiler.py (vc_major>8) 或 msvccompiler.py (低版本)。

3.1.3 解决方案分析

有了以上的准备知识，就能很容易理解之前的处理方法了。

“**Python 2.7使用的是MS 9.0编译器不支持C++11，然而，fasttext需要C++11的支持**”

既然默认的编译器不能够满足我们的需求，就去手动改掉python获取VC编译器的代码，直接返回用户需要的版本号来解决这个问题。虽然最后看上去很简单，但是从接触到顺利安装fasttext花费了我两天时间，希望这篇帖子能对大家有所帮助。