实验步骤

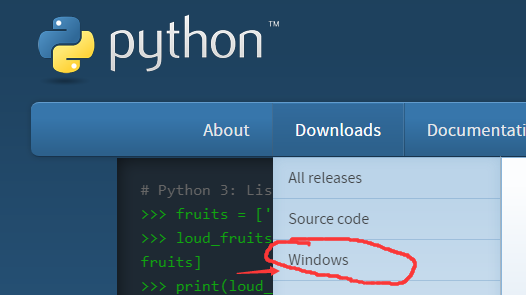
1. 实验环境搭建

由于实验代码使用的是python语言，因此首先需要安装python编译环境。我们在百度中直接搜索python，找到python官网，从官网中下载安装即可。具体的过程如下图：

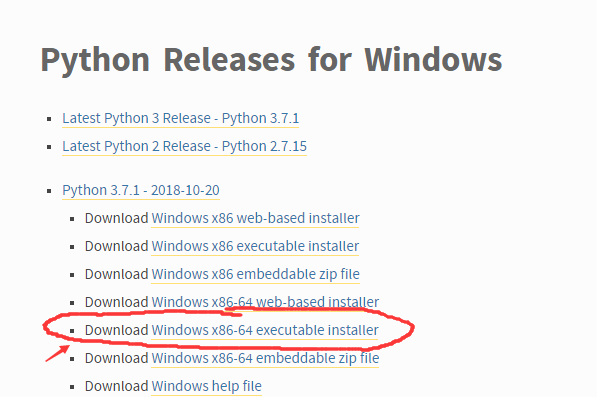
首先，搜索python，找到python官网。



进入官网后，点击download中windows条目，进入下载页面。



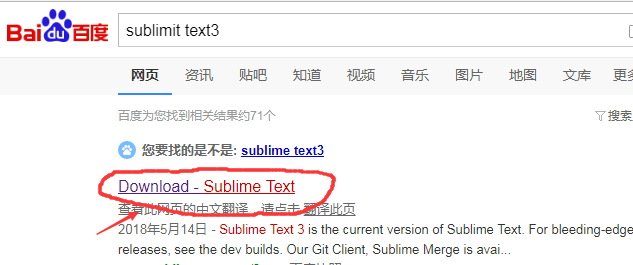
在windows操作系统下，可以下载的python版本很多，这里我们不一一列举，我们推荐大家直接下载最新更新的版本，并且直接下载可执行的安装版本，也就是一个exe程序，直接安装。因为我的机器是64位操作系统，因此我选择如下图所示的x86-64版本，如果您的机器还停留在32位，请对应的下载x86版本即可。



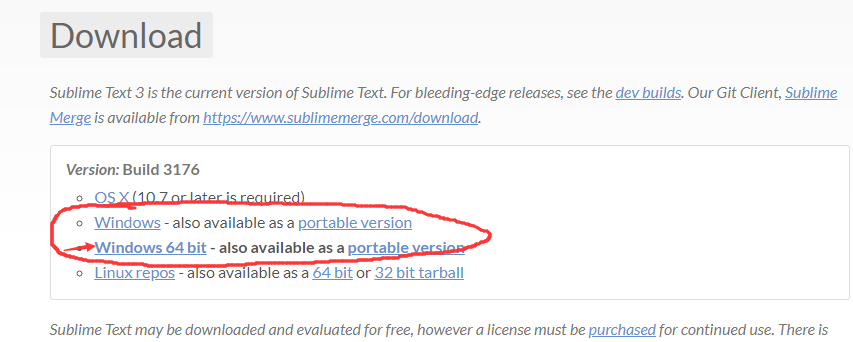
文件下载完成后，点击安装。

1. 编辑器下载

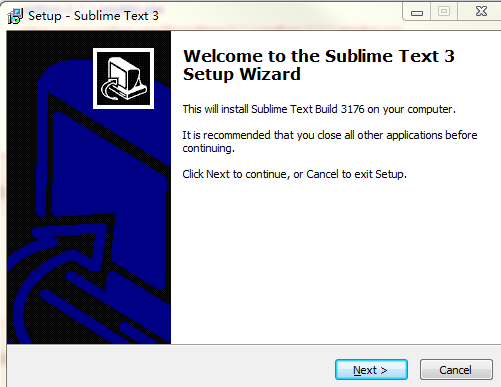
本人采用的是sublimi text编辑器编辑代码，同样我们从网上下载完安装到自己电脑上面即可。我们在百度中查找sublimi text3,点击进入网页下载，如下图，



进入该网页后，主界面就可以直接下载，对应的同样有不同的操作系统平台，这里我们选择windows 64位。



下载完成后，点击安装即可。安装程序运行如下图，我们如果不更改安装目录，只需要一直点击next，最后点击install即可。



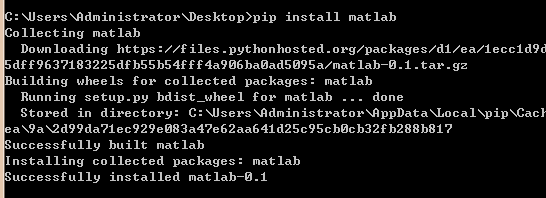
至此，实验环境已经全部搭建完毕，接下来只需要导入我们实验所需要的工具包或者库即可。

1. 实验库导入

本实验我们用到了gensim库，numpy库，nltk库，whoosh库，re库等等，导入这些库的操作很简单，这里使用pip install指令即可。先调出windows命令行，然后在命令行中执行pip指令。

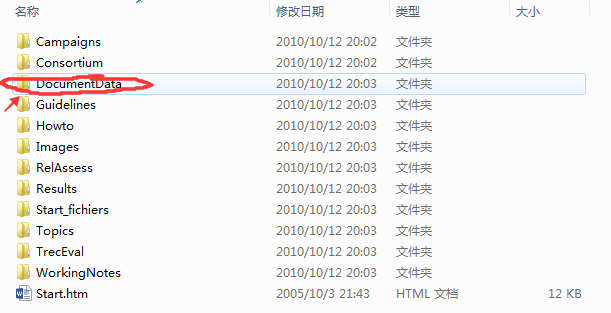
1. 在桌面上按住shift，同时点击鼠标右键，选择在此处打开命令窗口即可。
2. 在命令行中输入命令，例如：pip install genism
3. 一直等待命令提示安装成功，也就是显示success，那么导入库成功。

因为本机已经导入了上面的库，这里我们以没有导入过的matlab库为例子，展示如下图，

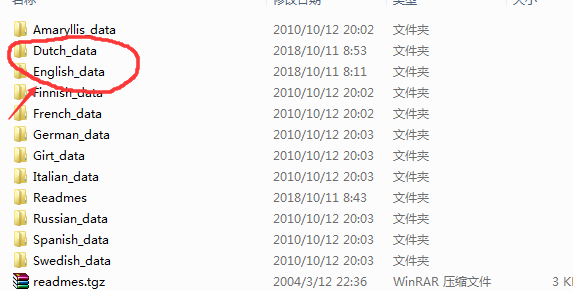


1. 实验步骤
2. **数据预处理**。

需要用的测试集和训练集以及对应的查询评估文件都已经上传到百度云，具体的目录如下：



测试集在我们上图中所标记的文件夹下，我们在对应的DataCollections下可以找到不同语言下的测试集，包括英文，荷兰语，西班牙等等。本实验我们主要用到的就是英文和荷兰语。



把上面标记的2个数据集作为本实验的训练集即可。获取到测试集后，下面就是需要我们在代码中运行，来对测试集进行处理。

测试集处理的代码是test\_collection\_preprocess.py，这里我们主要更改一下主程序中的文件路径就可以运行。



首先我们对英文测试集处理，英文测试集分为2个部分，一部分是gh95，另外一部分是la94，由于2个部分文本中处理方法不同，因此按照文件中提示我们要先处理gh95，然后再处理la94。

outdir代表处理完后输出的文本集合，代码中我设置的是相对路径名，这里可以根据自身需求设置到不同的路径下，更改成不同的文件名。

处理集合为英文，因此我们调用data\_process方法，参数为english，当处理集合为其它语言的时候，我们更换参数即可。

先处理gh95，这里调用getFiles获取文件夹下所有文件名，参数这里我们设置为gh95的根目录。也就是当我们把英文文档解压之后，我们会得到如下文件，



在GlasgowHeald95文件夹下，我们在进入下一级，会有一个文件夹和一个文件，

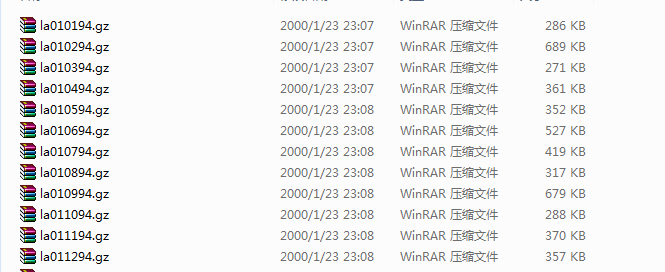


我们要处理的所有文档都包含在GH95中，因此getFiles方法中参数的路径就为GH95这个文件夹的绝对路径即可。

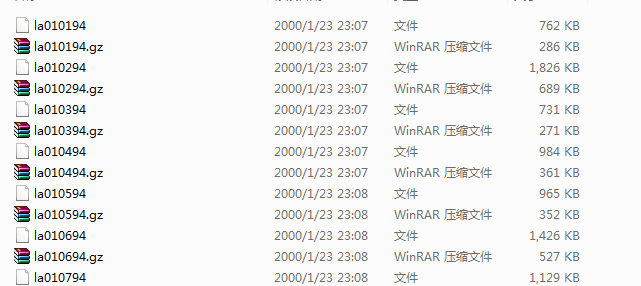
获取完文件名之后，我们调用对应的数据集处理方法即可。比如，现在我们处理的是英文文档GH95数据集，在类中我们定义的处理方法是GH\_per\_process，参数是输出文件路径，也就是我们刚刚开始定义的outdir。

在编辑器中运行一遍该代码，编辑器的运行快捷键是Ctrl+b。

同样，我们接着再处理LA94，这里前面的outdir不用变，只需要改变getFiles中的参数，因为要获取la文件夹下所有文档，所以需要换成la的文件路径，这里需要注意的是，在LATimes94下的所有文档仍然以压缩文件的形式出现，考虑到用python自带的读取压缩文件方法会出现乱码情况，因此我们需要手工利用解压软件对所有文档解压。具体情况如下，原本的文件夹下内容，



解压到当前文件夹下，如下图，



解压之后，我们的getFiles会剔除掉.gz，只读取解压后的文件名，接着执行LA\_pre\_process方法即可。outdir不用更换，和前面处理gh95一致，因为我们是在原来的文本基础上接着写入。

最后outdir即为我们处理完成后的英文文档集合。

在处理荷兰语的时候，我们需要将主程序中代码全部替换，首先outdir更改，这里为了方便，我只更改了文件名，例如将EN\_test\_collection.txt改为NL\_test\_collection.txt，类的初始化代码换为荷兰语data\_process(‘dutch’)，由于荷兰语中2个部分处理一致，这样我们调用a.getAllfile获取所有文件名，该方法中参数为荷兰语的根目录。例如，我们原本解压之后的文件，把它们放到Dutch\_data文件夹下，如下图所示。



我数据的存放路径如下，



那么方法中传入的路径名字就是’F:/CLEF2003/first test collection/Dutch\_data/’。

最后，我们调用荷兰语处理方法，更改为a.NL\_pre\_process(outdir)。

运行一遍代码，也会得到一个处理之后的荷兰语测试集。

当我们得到2个处理之后的测试集合之后，标志着我们的数据预处理部分已经完成。接下来根据测试集合，构建索引结构。

1. **构建索引**

构建索引，我们只要执行index.py中的代码即可。

里面的data\_dir就是第一步中我们处理后的文档，假如我们对英文文档构建索引，传入的就是英文文档路径，相反荷兰语文档就是荷兰语文档路径，前面我们建议将英文文档和荷兰语文档保存在一起，这里只要将前面的outdir直接copy过来就完事。

接下来，会选择创建index的目录位置，代码中是在当前目录下创建一个index文件夹，文件夹下面存放的是index，为了简便，我们创建一个index\_en,一个index\_nl,

**if** not os.path.exists("index\_en"):

os.mkdir("index\_en")

ix=create\_in("index\_en",schema)

执行一遍程序，生成英文索引，接着更改data\_dir为荷兰语集合，

**if** not os.path.exists("index\_nl"):

os.mkdir("index\_nl")

ix=create\_in("index\_nl",schema)

执行一遍程序，这样荷兰语的索引也构建完毕。

1. **查询生成**

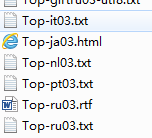
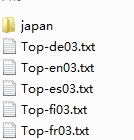
为了方便，我已经将最新的查询生成代码上传，代码做出的修改主要是将英文查询和荷兰语查询的生成整合到方法内部，用户不用考虑内部操作细节，具体的解析如下。

代码中我们需要执行的方法就是get\_query方法，参数依次输入文件路径，输出文件路径，查询语言，目标语言。

输入文件路径，就是topics文件夹下的文件路径，



我们可以看到有多个年份的topics文件夹，这里做哪一年的实验就对应选择哪一年的topics即可。



选择英文topics和荷兰语topics即可，这样输入文件路径就是这2个文件在本地的路径。

同样，输出文件路径可以自己选择，命名方式可以参照之前测试集的命名。

查询语言，代表查询语句的语种，目标语言，代表检索文档语种。比如，我做荷兰语单语实验，即输入荷兰语查询，检索荷兰语文档，也就是前面生成的测试集。那么参数分别为’dutch’,’dutch’，如果我做的是英文-荷兰语双语实验，即输入英文查询词，检索荷兰语文档，那么参数分别为’english’,’dutch’。

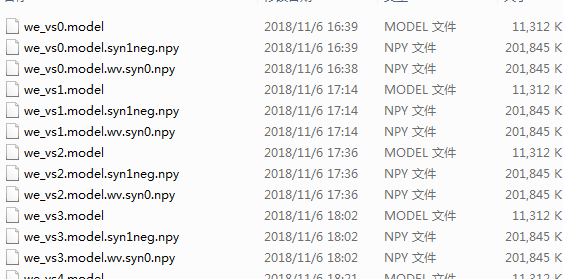
同理，英文单语和荷兰语-英文双语也是这么理解。

执行完成后，会有处理之后对应的查询文件生成。

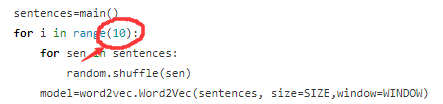
1. **词向量训练**

词向量的训练执行WE\_training.py即可，我们主要介绍几个需要修改的参数，一个是outdir，也就是词向量生成路径，建议创建一个空的文件夹，然后将该文件夹指定给outdir，另外一个是size，也就是训练词向量的维度，我们只考虑300维和600维，再一个就是main函数中的dir，是训练集的路径，训练集我将通过百度云分享，届时下载即可。

这里我展示一下生成的词向量文件夹下文件。



这只是一部分，因为我们默认生成10组，由于生成十组结果时间比较长，如果不想生成十组，那么可以将数字修改为3或者5也可以，依自身情况。



修改完成之后，执行一遍代码即可得到词向量模型。

1. **检索**

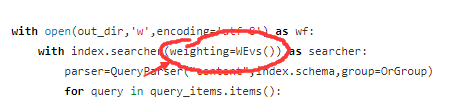
这里我们也只需要修改如下几个参数。

Index\_dir，代表索引存放路径，如果我们检索目标集合为英文文档，修改为英文索引路径，如果我们检索目标集合为荷兰语文档，那么修改为荷兰语索引路径。

Query\_dir,代表查询文件路径，也就是前面query\_generate.py生成的查询文件，这里我们建议在查询文件命名的过程中就将语言信息和单语还是双语信息包括在其中，比如代码中的单语荷兰语查询，又如，双语英文查询可以命名为bienglish,这样一看就知道是英文检索荷兰语的实验。

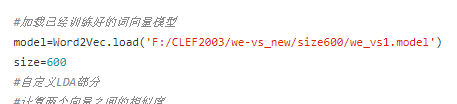
Outdir,结果文件存放路径，建议一个实验的结果全部放在一起，便于后面评估，单语english，单语dutch,双语en-dutch,双语dutch-en,这四个即可。

最后一个参数也就是权重模型参数，代码中要修改的部分如图所示，



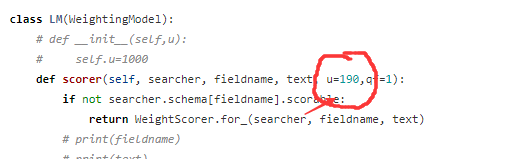
代码中我们使用的是自定义词向量模型计算向量相似度来评估查询和文档之间的联系，当使用该模型的时候，需要到模型中修改几个参数。下面我们来到WeightingModel.py。

如图所示，我们需要修改的部分，一个是加载训练好词向量的路径，这个路径也就是上面我们训练词向量生成的路径。另外一个是词向量的维度，如果我们加载的是600维，那么size设置为600，如果300维，size设置为300.



如果我们要使用其他模型方法，下面我说明一下几种常见的模型。

1. 语言模型，首先需要修改一句，将from WeightingModel import WEvs修改为from WeightingModel import WEvs,LM，意思就是从对应的文件模块中导入需要使用的类。然后就将weighting=WEvs()换成weighting=LM()。但是这里需要注意的是LM中存在一个名为u的参数，在WeightingModel中可以修改不同的u值。我这里u是190.



1. BM25，这里不需要修改参数，只要将Weighting=WEvs()去掉就行，因为默认的模型是BM25.
2. TFIDF,需要将Weighting=WEvs(),修改为Weighting=scoring.TF\_IDF().

最后，需要注意的是，输出文件名字一定要修改，每种对应的模型给一个对应的名字即可，不要用一样的outdir。比如代码中我们最开始是

out\_dir="F:/CLEF2003/result/monolingual/nl/pysize600.txt"

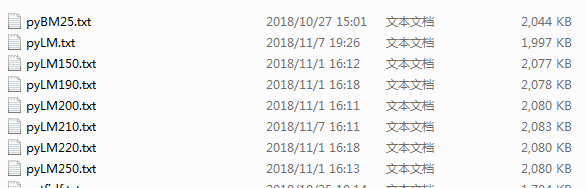
当我们用BM25模型，我们修改为

Outdir=’F:/CLEF2003/result/monolingual/nl/pyBM25.txt’

同理，LM和tfidf都可以这样修改。

1. **评估**

前面我们将检索结果存放在一个文件夹下，例如我的结果文件夹下。



在map.py中，evolution\_dir的值就设置为该文件夹，代表需要评估的文件目录。同时，access\_dir代表评估标准文件。



上图就是E0008中的评估标准文件。如果检索的是英文文档，就对应qrels\_EN,如果检索荷兰语文档，对应的就是qrels\_NL.

执行一遍程序，就可以得到评估结果。