1. 英文文本挖掘预处理特点

　　　　英文文本的预处理方法和中文的有部分区别。首先，英文文本挖掘预处理一般可以不做分词（特殊需求除外），而中文预处理分词是必不可少的一步。第二点，大部分英文文本都是uft-8的编码，这样在大多数时候处理的时候不用考虑编码转换的问题，而中文文本处理必须要处理unicode的编码问题。这两部分我们在中文文本挖掘预处理里已经讲了。

　　　　而英文文本的预处理也有自己特殊的地方，第三点就是拼写问题，很多时候，我们的预处理要包括拼写检查，比如“Helo World”这样的错误，我们不能在分析的时候讲错纠错。所以需要在预处理前加以纠正。第四点就是词干提取(stemming)和词形还原(lemmatization)。这个东西主要是英文有单数，复数和各种时态，导致一个词会有不同的形式。比如“countries”和"country"，"wolf"和"wolves"，我们期望是有一个词。

　　　　后面的预处理中，我们会重点讲述第三点和第四点的处理。

2. 英文文本挖掘预处理一：数据收集

　　　　这部分英文和中文类似。获取方法一般有两种：使用别人做好的语料库和自己用爬虫去在网上去爬自己的语料数据。

　　　　对于第一种方法，常用的文本语料库在网上有很多，如果大家只是学习，则可以直接下载下来使用，但如果是某些特殊主题的语料库，比如“deep learning”相关的语料库，则这种方法行不通，需要我们自己用第二种方法去获取。

　　　　对于第二种使用爬虫的方法，开源工具有很多，通用的爬虫我一般使用beautifulsoup。但是我们我们需要某些特殊的语料数据，比如上面提到的“deep learning”相关的语料库，则需要用主题爬虫（也叫聚焦爬虫）来完成。这个我一般使用ache。 ache允许我们用关键字或者一个分类算法模型来过滤出我们需要的主题语料，比较强大。

3. 英文文本挖掘预处理二：除去数据中非文本部分

　　　　这一步主要是针对我们用爬虫收集的语料数据，由于爬下来的内容中有很多html的一些标签，需要去掉。少量的非文本内容的可以直接用Python的正则表达式(re)删除, 复杂的则可以用beautifulsoup来去除。另外还有一些特殊的非英文字符(non-alpha),也可以用Python的正则表达式(re)删除。

4. 英文文本挖掘预处理三：拼写检查更正

　　　　由于英文文本中可能有拼写错误，因此一般需要进行拼写检查。如果确信我们分析的文本没有拼写问题，可以略去此步。

　　　　拼写检查，我们一般用pyenchant类库完成。pyenchant的安装很简单："pip install pyenchant"即可。

　　　　对于一段文本，我们可以用下面的方式去找出拼写错误：

from enchant.checker import SpellChecker

chkr = SpellChecker("en\_US")

chkr.set\_text("Many peope likee to watch In the Name of People.")

for err in chkr:

print "ERROR:", err.word

　　　　输出是：

ERROR: peope

ERROR: likee

　　　　找出错误后，我们可以自己来决定是否要改正。当然，我们也可以用pyenchant中的wxSpellCheckerDialog类来用对话框的形式来交互决定是忽略，改正还是全部改正文本中的错误拼写。大家感兴趣的话可以去研究pyenchant的官方文档。

5. 英文文本挖掘预处理四：词干提取(stemming)和词形还原(lemmatization)

　　　　词干提取(stemming)和词型还原(lemmatization)是英文文本预处理的特色。两者其实有共同点，即都是要找到词的原始形式。只不过词干提取(stemming)会更加激进一点，它在寻找词干的时候可以会得到不是词的词干。比如"imaging"的词干可能得到的是"imag", 并不是一个词。而词形还原则保守一些，它一般只对能够还原成一个正确的词的词进行处理。个人比较喜欢使用词型还原而不是词干提取。

　　　　在实际应用中，一般使用nltk来进行词干提取和词型还原。安装nltk也很简单，"pip install nltk"即可。只不过我们一般需要下载nltk的语料库，可以用下面的代码完成，nltk会弹出对话框选择要下载的内容。选择下载语料库就可以了。

import nltk

nltk.download()

　　　　在nltk中，做词干提取的方法有PorterStemmer，LancasterStemmer和SnowballStemmer。个人推荐使用SnowballStemmer。这个类可以处理很多种语言，当然，除了中文。

from nltk.stem import SnowballStemmer

stemmer = SnowballStemmer("english") # Choose a language

stemmer.stem("countries") # Stem a word

　　　　输出是"countri",这个词干并不是一个词。

　　　　而如果是做词型还原，则一般可以使用WordNetLemmatizer类，即wordnet词形还原方法。

from nltk.stem import WordNetLemmatizer

wnl = WordNetLemmatizer()

print(wnl.lemmatize('countries'))

　　　　输出是"country",比较符合需求。

　　　　在实际的英文文本挖掘预处理的时候，建议使用基于wordnet的词形还原就可以了。

　　　　在这里有个词干提取和词型还原的demo，如果是这块的新手可以去看看，上手很合适。

6. 英文文本挖掘预处理五：转化为小写

　　　　由于英文单词有大小写之分，我们期望统计时像“Home”和“home”是一个词。因此一般需要将所有的词都转化为小写。这个直接用python的API就可以搞定。

7. 英文文本挖掘预处理六：引入停用词

　　　　在英文文本中有很多无效的词，比如“a”，“to”，一些短词，还有一些标点符号，这些我们不想在文本分析的时候引入，因此需要去掉，这些词就是停用词。个人常用的英文停用词表下载地址在这。当然也有其他版本的停用词表，不过这个版本是我常用的。

　　　　在我们用scikit-learn做特征处理的时候，可以通过参数stop\_words来引入一个数组作为停用词表。这个方法和前文讲中文停用词的方法相同，这里就不写出代码，大家参考前文即可。

8. 英文文本挖掘预处理七：特征处理

　　　　现在我们就可以用scikit-learn来对我们的文本特征进行处理了，在文本挖掘预处理之向量化与Hash Trick中，我们讲到了两种特征处理的方法，向量化与Hash Trick。而向量化是最常用的方法，因为它可以接着进行TF-IDF的特征处理。在文本挖掘预处理之TF-IDF中，我们也讲到了TF-IDF特征处理的方法。

　　　　TfidfVectorizer类可以帮助我们完成向量化，TF-IDF和标准化三步。当然，还可以帮我们处理停用词。这部分工作和中文的特征处理也是完全相同的，大家参考前文即可。

9. 英文文本挖掘预处理八：建立分析模型

　　　　有了每段文本的TF-IDF的特征向量，我们就可以利用这些数据建立分类模型，或者聚类模型了，或者进行主题模型的分析。此时的分类聚类模型和之前讲的非自然语言处理的数据分析没有什么两样。因此对应的算法都可以直接使用。而主题模型是自然语言处理比较特殊的一块，这个我们后面再单独讲。

10. 英文文本挖掘预处理总结

　　　　上面我们对英文文本挖掘预处理的过程做了一个总结，希望可以帮助到大家。需要注意的是这个流程主要针对一些常用的文本挖掘，并使用了词袋模型，对于某一些自然语言处理的需求则流程需要修改。比如有时候需要做词性标注，而有时候我们也需要英文分词，比如得到"New York"而不是“New”和“York”，因此这个流程仅供自然语言处理入门者参考，我们可以根据我们的数据分析目的选择合适的预处理方法。