一、训练词向量

1、分布式词向量

Word2vec是Google在2013年提出的。Word2vec是一种神经网络实现，用于学习词的分布式表达（distributed representations for words）。

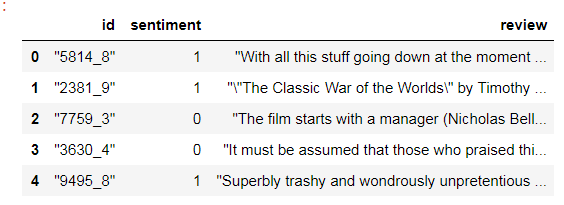
Word2vec即使不利用标签，也能产生有意义的表达。这是非常有用的，因为大部分真实世界里的数据是没有标签的。如果给的词足够多，词向量会展现很多有趣的特性。比如有相似意义的词会出现在一个类里，而不同的类是有间隔的，这种特性可以让词之间的关系，可以通过向量计算来表示。一个有名的例子是：king - man + woman = queen。

分布式词向量对于词预测和机器翻译很有用，这次我们用来做情感分析。

2、训练前的预备工作

首先，用pandas导入数据，我们使用IMDB的电影评论数据，这是一个通用的机器学习的数据集。里面包括25000条标注情感倾向的训练集和25000条标注了情感倾向的测试集以及50000条没有标注的数据。

我们看一下 labeledTrainData，里面包含了25000条IMDB电影评论，每一条评论都有一个表示情绪的正标签或负标签。



我们需要对所有的训练数据和测试数据进行清洗，包括去除html,符号，将段落分割成句子。因为输入word2vec里面的是一个句子。值得注意的是，在word2vec里面，为了尽可能保留词的上下文信息，我们不会删除停用词。

将所有的训练数据，包括有标签的训练集和没有标签的训练集，一起汇总到一个大的训练集。将这个训练数据利用word2vec进行训练。

3、训练词向量

Word2Vec的主要参数

# Set values for various parameters

Architecture:你算法的架构，你是选择skip-gram算法（默认的）还是选择CBOW算法。一般来说，skip-gram要比CBOW要慢，但是效果会好点，如果你的数据集不是非常的大，建议用ship-gram.

Training algorithm: 你训练词向量的算法。Hierarchical softmax or negative sampling. 本质上来，霍夫曼树和负采样都是用来加快训练速度的。默认都是用Hierarchical softmax。

Downsampling of frequent words: 常用词的下采样。你可以简单理解成把我们的语料中的常用词的权重降低。谷歌官方文档给的标准是0.00001 and 0.001. 我觉得差别不大。

Word vector dimensionality: 一般来说，词向量的维度越大，能够展示的一个词的上下文的信息越多，但是你需要同时考虑计算量和过拟合的问题。在本次训练中，我选的300.

Context / window size:在训练算法中，你的词窗口的大小。一般情况下取10.你也可以考虑取更大的窗口去尝试。

Worker threads: 程序占用的进程的个数。一般4到8个。如果是你自己电脑，你也可以让它全部占用，这会加快速度，但是电脑会很卡。

Minimum word count: 一篇文档中有意义的词至少要出现一定的次数，才是有意义的。一般情况下10-100次都是可以的。你需要在不断测试中找到合适值。

num\_features = 300 # 你选择的词向量的维度

min\_word\_count = 40 # 有意义词出现次数

num\_workers = 4 # 进程

context = 10 # 词窗口

downsampling = 1e-3 # 常用词的下采样

from gensim.models import word2vec

print("Training model...")

model = word2vec.Word2Vec(sentences, workers=num\_workers,

size=num\_features, min\_count = min\_word\_count,

window = context, sample = downsampling)

#保存模型

model\_name = "model/300features\_40minwords\_10context"

model.save(model\_name)

这样一个完整的词向量就训练好了。下一步，我们会利用训练好的词向量去进行文本的情感倾向的预测。