挖掘用户关注点：关键词提取与主题提取

一、文本关键词提取

> 文本关键词抽取，是对文本信息进行高度凝练的一种有效手段，通过几个词语准确概括文本的主题，帮助读者快速理解文本信息。目前，用于文本关键词提取的主要方法有四种：基于TF-IDF的关键词抽取、基于TextRank的关键词抽取、基于Word2Vec词聚类的关键词抽取以及多种算法相融合的关键词抽取。本文对于使用TF-IDF方法、TextRank方法和Word2Vec词聚类方法实现对于淘宝评论（同样适用于其它类型文本）关键词抽取的原理及流程进行了简单介绍，希望同学们能够通过理论与实践相结合的方式，一步步了解、学习、实现中文文本关键词抽取。

1 概述

一篇文档的关键词等同于最能表达文档主旨的N个词语，即对于文档来说最重要的词，因此，可以将文本关键词抽取问题转化为词语重要性排序问题，选取排名前TopN个词语作为文本关键词。目前，主流的文本关键词抽取方法主要有以下两大类：

（1）基于统计的关键词提取方法

该类方法根据统计信息，如词频，来计算得到文档中词语的权重，按权重值排序提取关键词。TF-IDF和TextRank均属于此类方法，其中TF-IDF方法通过计算单文本词频（Term Frequency， TF）和逆文本频率指数（Inverse Document Frequency， IDF）得到词语权重；TextRank方法基于PageRank的思想，通过词语共现窗口构建共现网络，计算词语得分。此类方法简单易行，适用性较强，然而未考虑词序问题。

（2）基于机器学习的关键词提取方法

该方法包括了SVM、朴素贝叶斯等有监督学习方法，以及K-means、层次聚类等无监督学习方法。在此类方法中，模型的好坏取决于特征提取，而深度学习正是特征提取的一种有效方式。由Google推出的Word2Vec词向量模型，是自然语言领域中具有代表性的学习工具。它在训练语言模型的过程中将词典映射到一个更抽象的向量空间中，每一个词语通过高维向量表示，该向量空间中两点之间的距离就对应两个词语的相似程度。

基于以上研究，可以采用**TF-IDF、TextRank和Word2Vec词聚类等方法**，利用Python语言进行开发，实现文本关键词的抽取。

2 开发环境准备

2.1 Python环境

2.2 第三方模块

本实验Python代码的实现使用到了多个著名的第三方模块，主要模块如下所示：

（1）Jieba

目前使用最为广泛的中文分词组件。

(2) Gensim

用于主题模型、文档索引和大型语料相似度索引的python库，主要用于自然语言处理（NLP）和信息检索（IR）。中文词向量模型构建需要用到该模块。

(3) Pandas

用于高效处理大型数据集、执行数据分析任务的python库，是基于Numpy的工具包。

(4) Numpy

用于存储和处理大型矩阵的工具包。

(5) Scikit-learn

用于机器学习的python工具包，python模块引用名字为sklearn，安装前还需要Numpy和Scipy两个Python库。

本实例中主要用到了该模块中的feature_extraction、聚类算法和PCA（pac降维算法）。

(6) Matplotlib

Matplotlib是一个python的图形框架，用于绘制二维图形。

3 数据准备

3.1 样本语料

淘宝平台中三个品类商品的用户评论数据，涉及服装、食品及化妆品，见附件。

3.2 停用词词典

使用中科院计算所中文自然语言处理开放平台发布的中文停用词表，包含了1208个停用词。下载地址：http://www.hicode.cc/download/view-software-13784.html

如需人工新增停用词，可直接在停用词表中添加，一行为一个停用词。

4 基于TF-IDF的文本关键词抽取方法

4.1 TF-IDF算法思想

词频（Term Frequency，TF）指某一给定词语在当前文件中出现的频率。由于同一个词语在长文档中可能比短文档有更高的词频，因此根据文档的长度，需要对给定词语进行归一化，即用给定词语的次数除以当前文件的总词数。

逆向文档频率（Inverse Document Frequency，IDF）是一个词语普遍重要性的度量。即如果一个词语只在很少的文档中出现，表示更能代表文档的主旨，它的权重也就越大；如果一个词在大量文件中都出现，表示不清楚代表什么内容，它的权重就应该小。

TF-IDF的主要思想是，如果某个词语在一篇文章中出现的频率高，并且在其他文章中较少出现，则认为该词语能较好的代表当前文章的含义。即一个词语的重要性与它在文档中出现的次数成正比，与它在语料库中文档出现的频率成反比。

计算公式如下：

$$词频（TF）=\frac{词w在文档中出现的次数}{文档的总词数}$$

$$逆文档频率（IDF）=\log（\frac{语料库的文档总数}{包含词w的文档数+1}）$$

$$TF\text{-}IDF=TF*IDF$$

4.2 TF-IDF文本关键词抽取方法流程

由以上可知，TF-IDF是对文本所有候选关键词进行加权处理，根据权值对关键词进行排序。假设D_n为测试语料的大小，该算法的关键词抽取步骤如下所示：

（1） 对于给定的文本D进行分词、词性标注和去除停用词等数据预处理操作。本文针对评论文本特征，选取部分词性的词作为候选关键词（也可不进行词性分析或自行选取部分词性）。采用结巴分词，保留'n','nz','v','vd','vn','l','a','d'这几个词性的词语，最终得到n个候选关键词，即D=[t1,t2,...,tn] ；

（2） 计算词语t_i 在文本D中的词频；

（3） 计算词语t_i 在整个语料的IDF=log (D_n /(D_t +1))，D_t 为语料库中词语t_i 出现的文档个数；

（4） 计算得到词语t_i 的TF-IDF=TF*IDF，并重复（2）—（4）得到所有候选关键词的TF-IDF数值；

（5） 对候选关键词计算结果进行倒序排列，得到排名前TopN个词汇作为关键词。

4.3 代码实现

Python第三方工具包Scikit-learn提供了TFIDF算法的相关函数，本文主要用到了sklearn.feature_extraction.text下的TfidfTransformer和CountVectorizer函数。其中，CountVectorizer函数用来构建语料库的中的词频矩阵，TfidfTransformer函数用来计算词语的tfidf权值。

注：TfidfTransformer()函数有一个参数smooth_idf，默认值是True，若设置为False，则IDF的计算公式为idf=log(D_n /D_t) + 1。

基于TF-IDF方法实现文本关键词抽取的代码执行步骤如下：

（1）读取样本源文件;

（2）加载自定义停用词表stopWord.txt，并对拼接的文本进行数据预处理操作，包括分词、筛选出符合词性的词语、去停用词，用空格分隔拼接成文本;

（3）遍历文本记录，将预处理完成的文本放入文档集corpus中；

（4）使用CountVectorizer()函数得到词频矩阵，a[j][i]表示第j个词在第i篇文档中的词频；

（5）使用TfidfTransformer()函数计算每个词的tf-idf权值；

（6）得到词袋模型中的关键词以及对应的tf-idf矩阵；

（7）遍历tf-idf矩阵，打印每篇文档的词汇以及对应的权重；

（8）对每篇文档，按照词语权重值降序排列，选取排名前topN个词最为文本关键词，并写入数据框中；

（9）将最终结果写入文件keys_TFIDF.csv中。

5 基于TextRank的文本关键词抽取方法

5.1 PageRank算法思想

TextRank算法是基于PageRank算法的，因此，在介绍TextRank前不得不了解一下PageRank算法。

PageRank算法是Google的创始人拉里·佩奇和谢尔盖·布林于1998年在斯坦福大学读研究生期间发明的，是用于根据网页间相互的超链接来计算网页重要性的技术。该算法借鉴了学术界评判学术论文重要性的方法，即查看论文的被引用次数。基于以上想法，PageRank算法的核心思想是，认为网页重要性由两部分组成：

① 如果一个网页被大量其他网页链接到说明这个网页比较重要，即被链接网页的数量；

② 如果一个网页被排名很高的网页链接说明这个网页比较重要，即被链接网页的权重。

一般情况下，一个网页的PageRank值（PR）计算公式如下所示：

$$\mathrm{PR}(p_i) = \frac{1-\alpha}{N} + \alpha \sum_{p_j \in M_{pi}} \frac{PR(p_j)}{L(p_j)}$$

其中，PR(Pi)是第i个网页的重要性排名即PR值；α是阻尼系数，一般设置为0.85；N是网页总数；Mpi 是所有对第i个网页有出链的网页集合；L(Pj)是第j 个网页的出链数目。

初始时，假设所有网页的排名都是1/N，根据上述公式计算出每个网页的PR值，在不断迭代趋于平稳的时候，停止迭代运算，得到最终结果。一般来讲，只要10次左右的迭代基本上就收敛了。

5.2 TextRank算法思想

TextRank算法是Mihalcea和Tarau于2004年在研究自动摘要提取过程中所提出来的，在PageRank算法的思路上做了改进。该算法把文本拆分成词汇作为网络节点，组成词汇网络图模型，将词语间的相似关系看成是一种推荐或投票关系，使其可以计算每一个词语的重要性。

基于TextRank的文本关键词抽取是利用局部词汇关系，即共现窗口，对候选关键词进行排序，该方法的步骤如下：

（1） 对于给定的文本D进行分词、词性标注和去除停用词等数据预处理操作。本文针对评论文本特征，选取部分词性的词作为候选关键词（也可不进行词性分析或自行选取部分词性）。采用结巴分词，保留'n','nz','v','vd','vn','l','a','d'这几个词性的词语，最终得到n个候选关键词，即D=[t1,t2,...,tn] ；

（2） 构建候选关键词图G=(V,E)，其中V为节点集，由候选关键词组成，并采用共现关系构造任两点之间的边，两个节点之间仅当它们对应的词汇在长度为K的窗口中共现则存在边，K表示窗口大小即最多共现K个词汇；

（3） 根据公式迭代计算各节点的权重，直至收敛；

（4） 对节点权重进行倒序排列，得到排名前TopN个词汇作为关键词。

说明：Jieba库中包含jieba.analyse.textrank函数可直接实现TextRank算法，可采用该函数进行实验。

5.3 代码实现

基于TextRank方法实现文本关键词抽取的代码执行步骤如下：

（1）读取样本源文件;

（2）加载停用词表stopWord.txt;

（3）遍历文本记录，采用jieba.analyse.textrank函数筛选出指定词性，以及topN个文本关键词，并将结果存入数据框中；

（4）将最终结果写入文件keys_TextRank.csv中。

6 基于Word2Vec词聚类的文本关键词抽取方法

6.1 Word2Vec词向量表示

众所周知，机器学习模型的输入必须是数值型数据，文本无法直接作为模型的输入，需要首先将其转化成数学形式。基于Word2Vec词聚类方法正是一种机器学习方法，需要将候选关键词进行向量化表示，因此要先构建Word2Vec词向量模型，从而抽取出候选关键词的词向量。

Word2Vec是当时在Google任职的Mikolov等人于2013年发布的一款词向量训练工具，一经发布便在自然语言处理领域得到了广泛的应用。该工具利用浅层神经网络模型自动学习词语在语料库中的出现情况，把词语嵌入到一个高维的空间中，通常在100-500维，在新的高维空间中词语被表示为词向量的形式。与传统的文本表示方式相比，Word2Vec生成的词向量表示，词语之间的语义关系在高维空间中得到了较好的体现，即语义相近的词语在高维空间中的距离更近；同时，使用词向量避免了词语表示的“维度灾难”问题。

就实际操作而言，特征词向量的抽取是基于已经训练好的词向量模型，词向量模型的训练需要海量的语料才能达到较好的效果，抽取候选关键词的词向量作为聚类模型的输入。

另外，在阅读资料的过程中发现，有些十分专业或者生僻的词语可能wiki中文语料中并未包含，为了提高语料的质量，可新增实验所需的样本语料一起训练。本例中为了简便并未采取这种方法，同学们可参考此种方法根据自己的实际情况进行调整。

此外，BERT等预训练模型当前被广泛用于词向量表示，并可以通过微调较好完成自然语言处理的多项任务，同学们也可以自行尝试。

6.2 聚类算法

聚类算法旨在数据中发现数据对象之间的关系，将数据进行分组，使得组内的相似性尽可能的大，组件的相似性尽可能小。由于上次作业已涉及相关方法介绍，此处不再赘述。

6.3 Word2Vec词聚类文本关键词抽取方法流程

Word2Vec词聚类文本关键词抽取方法的主要思路是对于用词向量表示的文本词语，通过对文档词进行聚类，选择聚类中心作为一个主要关键词，计算其他词与聚类中心的距离即相似度，选择topN个距离聚类中心最近的词作为文本关键词，而这个词间相似度可用Word2Vec生成的向量计算得到。

假设D_n为测试语料的大小，使用该方法进行文本关键词抽取的步骤如下所示：

（1） 得到词向量文件；

（2） 对于给定的文本D进行分词、词性标注、去重和去除停用词等数据预处理操作。采用结巴分词，保留指定词性的词语，最终得到n个候选关键词，即D=[t1,t2,...,tn]；

（3） 遍历候选关键词，从词向量文件中抽取候选关键词的词向量表示，即WV=[v_1，v_2，…，v_m]；

（4） 对候选关键词进行聚类，得到各个类别的聚类中心；

（5） 计算各类别下，组内词语与聚类中心的距离（欧几里得距离），按聚类大小进行升序排序；

（6） 对候选关键词计算结果得到排名前TopN个词汇作为文本关键词。

步骤（4）中需要人为给定聚类的个数；步骤（5）中计算各词语与聚类中心的距离，常见的方法有欧式距离和曼哈顿距离，本文采用的是欧式距离，计算公式如下：

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$$

6.4 代码实现

Python第三方工具包Scikit-learn提供了一些聚类算法的相关函数，sklearn.decomposition.PCA()函数可用于数据降维以便绘制图形。

基于Word2Vec词聚类方法实现文本关键词抽取的代码执行步骤如下：

（1）读取样本源文件sample_data.csv;

（2）加载自定义停用词表stopWord.txt，并对拼接的文本进行数据预处理操作，包括分词、筛选出符合词性的词语、去重、去停用词，形成列表存储；

（3）读取词向量模型文件，从中抽取出所有候选关键词的词向量表示，存入文件中；

（4）读取文本的词向量表示文件，得到聚类结果以及聚类中心的向量表示；

（5）采用欧式距离计算方法，计算得到每个词语与聚类中心的距离；

（6）按照得到的距离升序排列，选取排名前topN个词作为文本关键词，并写入数据框中；

（7）将最终结果写入文件keys_word2vec.csv中。

7 作业提示

本文总结了三种常用的抽取文本关键词的方法：TF-IDF、TextRank和Word2Vec词向量聚类，仅作为作业参考，其中某些细节仍然可以进行改进。

例如，尝试TextRank等算法的改进方法；可在word2vec模型训练的原始语料之外加入相应的网络用语文本语料；尝试BERT等其他预训练模型生成词向量；去除所有文档中都包含的某一出现频次超过指定阈值的词语等。

同学们可根据自己的实际情况或者参考论文资料进行参数的优化以及细节的调整。

二、文本主题提取

> 主题是一个词项的集合，这些词项以高的概率事件去描述文本中所谈论的某一方面的内容。在文档层面，理解文本最有效的方式之一就是分析其主题。
>
> 在文档集合中学习、识别和提取这些主题的过程被称为主题建模。主题建模的技术从 LSA 发展到 PLSA，再到当前被广泛使用的LDA以及基于深度学习的lda2vec等。

1 主题模型概述

所有主题模型都基于相同的基本假设：

- 每个文档包含多个主题；
- 每个主题包含多个单词。

换句话说，主题模型围绕着以下观点构建：实际上，文档的语义由一些我们所忽视的隐变量或「潜」变量管理。因此，主题建模的目标就是揭示这些潜在变量——也就是主题，正是它们塑造了我们文档和语料库的含义。

通过本次作业，同学们可以学习了解不同种类的主题模型如何揭示这些潜在的主题。

2 LDA主题提取

LDA是主题模型中非常优雅的一个模型，自2003年，学术界输出了很多基于主题模型的变体，而要理解这些模型，首先需要理解LDA模型。由于该模型涉及较多数学知识，包括矩阵分解知识、二项分布、多项式分布、beta分布、狄利克雷分布、共轭先验概率分布等，以下文章对于模型原理及其代码实现有较为详细的介绍，同学们可以参考阅读。

模型原理

【NLP】LDA主题模型详解 https://blog.csdn.net/Daycym/article/details/88876460?utm_medium=distribute.pc_relevant.none-task-blog-2%7Edefault%7EBlogCommendFromMachineLearnPai2%7Edefault-12.control&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2%7Edefault%7EBlogCommendFromMachineLearnPai2%7Edefault-12.contro

代码实现

【NLP】LDA主题模型的简单应用 https://blog.csdn.net/Daycym/article/details/88935149

3 作业提示

对于淘宝评论数据，将每一条用户评论看作一个文档，所有用户提问即为文档集合，我们需要通过主题模型，得到该文档集合中所描述的主题，分析用户的关注点。

LDA模型是一个词袋模型，缺乏语序、语法、语义等高级特征的考虑，同学们可以针对本任务，尝试lda2vec等模型，或在LDA等基础模型的基础上进行改进。