第5章

文本聚类技术

5.1 概述

随着因特网的迅猛发展和广泛应用，人们已经跨进信息时代，人们生活和工作的方方面面都在因此发生着巨大的变化。当前网络上普遍存在着“信息爆炸”的问题，电子商务、微博、网络新闻、电子文档、E-mail、电子期刊以及网上书刊等在线信息日益增多，其中半结构化信息占据了很大一部分，甚至有部分信息是非结构化的。如何快速有效地处理因特网上这些令人挠头的海量信息，从中抽取出有用信息，是当前研究人员迫切想要解决的问题。人工分类这一传统做法，虽然在某些程度上有效地获取了信息，但是该做法过于费时费力。面对海量的网络信息，人工分类的处理方式被淘汰出局。现在的网络信息很多都使用无效标记，更有甚者没有标记，人工分类的方法根本无法适用。怎样在没有类别信息指导的前提下对网络文本进行分类并标识，为越来越多的研究人员所关注。聚类是一种无监督的机器学习方法，与有监督机器学习不同，它是一种完全自动化地处理文本的技术，不再需要人工地参与来辨别训练文档的类别，因此聚类方法也具有一定的灵活性，是组织文本信息的一种重要手段。

文本聚类是将文本数据集按照定义的文本相似度量函数分为若干文本子集的过程，并标注出每个文本子集的类别标签，其依据的假设是同类中的对象互相之间是相似的，而不同类中对象之间不具有相似性或具有很小的相似性。文本的聚类和分类是不相同的，由于分类有训练的过程，因此它可以在文本内容分析之后，按照预先制定的类别信息给该文本分配适宜的类别。聚类技术能够分析无类别标记的文本集，依据文本集的构造发现当中隐藏的类别信息。对文本进行分析并标注其类别，这样做有助于计算机识别文本集的内部信息，因此能够作为文档自动摘要、语义消歧等自然语言处理技术的预处理操作。

除此之外，聚类技术还可以改善分类结果，提高检索系统的性能，进而有助于提高用户查找信息的时效和功效。对于当下盛行的信息推荐（或服务推荐），聚类方法也能助其一臂之力。聚类方法主要是通过聚类分析用户频繁浏览的文档，发现文档规律中隐藏的用户兴趣模式来完成这个功能的。它还可以用于数字图书馆服务与文档集合的自动整理，用于流行串预警和热点主题辨别，及时发现网络热点话题并跟进话题的趋势动向，自动辨识网络上疯狂传播的木马特征，预见系统漏洞和黑客攻击的危险性，对国家建设的长治久安和社会生活的和谐发展意义非同一般。

相对而言，中文的结构和语义相当复杂，而英文的复杂度则小很多，但是中文在理论研究成果上还相当匮乏。随着因特网在中国的盛行，中文网络信息犹如雨后春笋层出不穷，中文发挥的作用不可低估，然而先前有效信息的获取途径却不再适用。因此，搞好中文文本聚类技术的研究，提高中文文本的自动化处理能力，具有重大的实际意义。

在向量空间模型中，用词空间中的一个向量来表示一篇文档，而一篇文档至少包含几千个词，因此文本聚类存在相当大的困难。

（1）高维性“维度灾难”的概念是由 Bellman 提出的，它的含义是，对于一个有很多变量的函数而言，因为随着数据对象属性维数的不断增加，网格单元的数量也会以指数级的速度增加，因此要在一个多维网格中去优化这个函数是不可能的事情。现在通常用“维度灾难”来代表在数据分析领域中因为变量过多而产生的各种问题。针对高维数据而言，如果将数据对象的每一个属性维度都当作一个变量，那么高维数据聚类问题就是一个典型的多变量下优化求解的问题，即“维度灾难”问题。高维数据聚类中的“维度灾难”问题除了会造成现有传统聚类算法效率低下外，对索引结构也会有很大的影响。另外，在高维空间中，查询点与它的最近邻点和最远邻点之间的距离在多数情况下是近似相等的，此时最近邻的概念不再有意义。

（2）稀疏性研究表明，数据对象在高维空间中的分布是非常稀疏的。假设数据集 D 的维数是 k，数据 k 在维空间中均匀分布，同时维和维之间是相互独立的，可以形象地认为数据集 D 存在于一个超立方体单元 $\Omega=[0.1]^k$ 中。假设一个超立方体的边长是 d 的值小于 1，则一个数据对象落在这个超立方体内的概率是 d^k。显而易见，d^k 的值非常小，并且随着数据对象属性维数的增加，d^k 的值会更小，那么在这个立方体中存在数据点的可能性也会更小。当数据对象的属性维数非常高时，在一个范围足够大的高维空间内极有可能不包含任何一个数据对象。例如，在 100 维的高维空间中，在一个边长等于 0.93 的超立方体中最多包含一个数据对象的概率仅为 0.0007。

（3）语义问题在中文文本中经常出现一词多义或一义多词的现象，这就导致近义词或同义词在文本中的出现是不可避免的。由于计算机本身并不能够识别文本的语义信息，这就使得文本机器聚类的结果和文本的实际聚类结果之间存在一定的差距。有研究发现，潜在语义索引方法能够加强文本间的语义关联，减少特征子集的维数，有效减少文本聚类的时间消耗。

5.2 常用的聚类方法

众所周知，我们所处的信息世界中广泛分布着各种类型的数据，显然，其中也不乏一些数量巨大（例如文本数据）以及维度高的数据集。有些对象需要成百上千个属性来描述，例如文本文档、分子生物数据、CAD 数据以及图像识别（图像数据是一个高维数据对象）、模式分类等。

很多聚类方法在统计学领域以及数据挖掘研究领域相继提出。但是应对数量巨大、维度超高的数据集的研究依然是当下聚类分析过程的热门所在。一般情况下，一个聚类算法的好坏依据已不仅仅是时效性高，而是各类数据集所表现的综合性能，如聚类质量、聚类速度、抑制噪声性能这些重要指标。一个较好的聚类算法通常拥有如下特性。

（1）具有不依靠任何领域知识却能够提供初始输入参数的值。

（2）可以发现任意形状的数据聚类。

（3）对高维空间以及海量数据集能够有效应对。

当前主流的聚类算法可以分为如下几类：基于划分的聚类方法，基于分层的聚类方法，基于密度的聚类方法，基于网格的聚类方法，以及基于模型的聚类方法等。比较著名的算法有 K-means 方法、K-中值方法、Birch 方法、DBSCAN 方法、STING 方法以及波形聚类方法。这些方法被广泛运用在各个科学领域，如图像遥感、数据噪声过滤、离群数据检测以及无监督的机器学习方式的文本聚类等。

5.2.1 基于划分的聚类方法

基于划分的聚类方法通常是将一个含有大量数据对象的数据集（数据对象数 n），通过划分方法将其划分成 m 个聚类，这样每个数据对象分区都代表一个类簇，同时 $m \leqslant n$；通过划分聚类方法可以把数据集划分成 m 个数据组，这 m 个数据组通常具有如下特点。

（1）每个类簇至少含有一个数据对象。

（2）每一个数据对象都属于确定的类簇。

但是，并不是每一种划分聚类方法都具有第二条特点，尤其是一些模糊划分类型的聚类算法，它的聚类结果中可能还有一些数据对象未被包含在类簇中，这些对象可能是噪声数据，也可能不是。通常情况下，对于基于划分的聚类方法需要给定参数 m 来初始化分区，m 同时代表的是数据集最终将被划分成类簇的个数。基于划分的聚类方法一般采用迭代的方法对数据集重复计算，以达到在各分布中将满足条件的数据对象从原来的类簇中挪到新的分类中，最终聚类结果满足条件收敛。评价划分聚类方法是否高效，一个重要的条件就是评判聚类结果量，如果满足在同一类簇中的对象相似性高，且明显区别于异簇对象时，那么这个划分方法就比较好。就目前来讲，比较经典的基于划分的聚类算法有 K-means 算法和 K-中值算法。其中，K-means 算法的聚类结果中，每个聚类代表着这个类簇中包含的全部数据对象在特定方法（距离方法）下计算得到结果均值；而对 K-中值算法来讲，其聚类结果中的类簇由每个类簇中距离类中心最近的数据对象代表。对于经典的基于划分的聚类算法来讲，其思路简单，容易发现各种规则的聚类，然而在大数据时

代，这些方法面对数据量大、形状不规整的对象集不能妥善处理。研究者们为应对以上情况，提出了许多改进方法以及混合聚类算法以弥补经典划分方法的不足。

5.2.2 基于分层的聚类方法

基于分层的聚类方法是将指定的数据集层次分解成多个对象聚类。层次聚类算法主要分成两类：凝聚聚类方法（自下而上的方式合并距离最近的相邻类簇）和分裂聚类方法（自上而下的方式将聚类分裂成独立的集群）。其中，凝聚聚类方法（AHC）是将数据集的每一个单独的模式合并成一个聚类，对于这个最终的聚类来讲，其只有一种模式，在这个过程中两个相邻最近的聚类合并成一个新的分组直到聚类包含所有模式为止。而对分裂聚类方法来讲，一个包含所有模式的聚类在初始时刻被创建，接下来这个聚类将被分解成两个类簇，这个过程直到各类簇被分解成的聚类只含有一个模式为止。其实对层次聚类方法来讲，就是将聚类初始阶段用户输入的各种模式通过计算输出成聚类形式的最终结果，通常情况下，聚类结果可以由树状图表示。

基于分层的聚类算法也存在如下缺点。首先，不论是凝聚方法还是分裂方法，一个步骤一旦开始了，就无法取消或是改变，显然这样对算法的时间消耗方面具有很大贡献，不用中途再做决策，但是对聚类质量来讲，一旦聚类过程出现问题，方法无法做出调整。基于以上缺陷，研究者们提出了不少相关改进算法，例如，CURE 算法，其充分考虑了分层中各分区对象之间的联系；BIRCH 算法，首先对数据集采用分层凝聚聚类以及迭代重定位。其次，对第一步产生的结果加以合并，并对其进行迭代和重新定位。

5.2.3 基于密度的聚类方法

通常情况下，绝大多数基于分层或划分的聚类算法都是通过一般的距离方法来计算两个对象之间的距离，并由此形成类簇，这类方法的优点就是方法简单、易于运用，但是它们只能发现有规则的球形聚类，而无法发掘其他形状的聚类，这也是这类方法的一个局限性。另外，基于划分的聚类算法在面对数据集中的噪声数据时处理效果较差，而基于密度的聚类算法对噪声点应对效果较好并对噪声数据不敏感。

很多新提出的聚类方法都结合了密度聚类方法的思想以吸纳其部分优点，密度聚类算法通常是衡量数据集中的相邻对象构成簇的密度，当数据密度超过了事先给定的阈值，则可以判定这些数据是在一个类簇之中，当然一般情况下都会给类簇设定一个最少包含对象数据的阈值，只有当类簇满足类簇密度以及最少包含对象数这两个阈值时，其才能被确定为一个类簇。正是因为如上特点，基于密度的聚类方法较其他密度聚类算法具有对噪声数据不敏感和可以发现任意形状的类簇的优势。在密度聚类方法中较为著名的是 DBSCAN 算法以及 OPTICS 算法。

但是密度聚类算法也有不少缺点。首先，当数据分布比较稀疏离散时，其聚类效果会比较差；其次，当数据量比较大时内存等相关硬件消耗过大；最后，聚类最少包含对象数（Minpts）以及扫描半径（Eps）这两个输入参数选择是否恰当关系到聚类的最终质量。

5.2.4 基于网格的聚类方法

基于网格的聚类方法主要思想就是将数据对象集的空间量化，并分配到有限的空间中，这些空间形成一个网状结构，并将聚类方法运用到网格结构中的对象集上。该方法相对于其他聚类算法的优势就是处理数度较快，并且在应对较大数据集的情况下，通常不会因为数据量的大小而产生性能上的大幅改变，依旧保持较好的数据处理性能。一般来讲，限制该类算法的主要因素是数据网格的数量。较为著名的网格聚类方法有 STING 以及 CLIQUE 方法，研究者们也提出了不少关于它们的拓展方法，但是对于基于网格的聚类算法来讲，其应对高维数据集的能力不足。

5.2.5 基于模型的聚类方法

模型聚类方法主要是对数据对象的每个聚类提出一个假设模型，并发掘出最匹配、最有效的模型来适配每个聚类。模型聚类方法通常采用一些数学函数（例如密度函数）来反映数据集的空间分布状态，并以此来设定数据聚类。这类聚类方法通常可以自学习地产生特定数量的聚类，以及合理应对噪声数据和利群点，旨在构建一个健壮的聚类方法。

此外，还有一些聚类算法将以上所涉及的几种聚类算法合理整合起来，如文献，使用两种或两种以上的聚类方法对数据集进行聚类。

具体的聚类方法优缺点如表 5-1 所示。

表 5-1 聚类方法特点比较

聚类方法名称	特　点	簇 特 征	优　点	缺　点
基于划分的聚类方法	对输入数据的顺序无特定要求，主要应对数值型数据	规整的球型，各聚类大小相近	方法较高效，简单易用	需对数据集多次扫描，对聚类初始条件敏感
基于层次的聚类方法	对数聚类型以及输入顺序无特殊要求	聚类形状不固定	应对大规模数据能力强，具备较强抗干扰能力	需要多次扫描数据集
基于密度的聚类方法	对数据类型以及输入顺序无特定要求	发现聚类形状的聚类	只需要扫描一次数据集，并能较强的应对噪声数据	应对高维数据能力较弱，依赖聚类参数
基于网格的聚类方法	对数据类型以及输入顺序无特定要求	发现聚类形状的聚类	速度较快. 仅需要一次扫描数据集	应对高维数据能力较弱
基于模型的聚类方法	对数据类型以及输入顺序无特定要求	发现聚类形状的聚类	应对噪声数据性能较强，聚类质量较高	对初始参数敏感，需要多次迭代

5.3 聚类算法的评价标准

聚类分析是一个富有挑战的研究领域，有关每一个应用都提出了一个自己独特的要求。以下就是对数据挖掘中的聚类分析的一些典型要求。

(1) 可扩展性。许多聚类算法在小数据集(少于 200 个数据对象)时可以工作很好,但一个大数据库可能会包含数以百万的对象,利用采样方法进行聚类分析可能得到一个有偏差的结果,这时就需要可扩展的聚类分析算法。

(2) 处理不同类型属性的能力。许多算法是针对基于区间的数值属性而设计的。但是有些应用需要针对其他类型数据,如二值类型、符号类型、顺序类型,或这些数据类型的组合。

(3) 发现任意形状的聚类。许多聚类算法是根据欧氏距离和 Manhattan 距离来进行聚类的。基于这类距离的聚类方法一般只能发现具有类似大小和密度的圆形或球状聚类。而实际上一个聚类是可以具有任意形状的,因此设计出能够发现任意形状类集的聚类算法是非常重要的。

(4) 需要(由用户)决定的输入参数最少。许多聚类算法需要用户输入聚类分析中所需要的一些参数(如期望所获聚类的个数)。聚类结果通常都与输入参数密切相关,而这些参数常常也很难决定,特别是包含高维对象的数据集。这不仅造成了用户的负担,而且也使得聚类质量难以控制。

(5) 处理噪声数据的能力。大多数现实世界的数据库均包含异常数据、不明确数据、数据丢失和噪声数据,有些聚类算法对这样的数据非常敏感并会导致获得质量较差的聚类结果。

(6) 对输入记录顺序不敏感。一些聚类算法对输入数据的顺序敏感也就是不同的数据输入会导致获得非常不同的结果。因此设计对输入数据顺序不敏感的聚类算法也是非常重要的。

(7) 高维问题。一个数据库或一个数据仓库或许包含若干维或属性。许多聚类算法在处理低维数据时(仅包含两三个维)时表现很好。人的视觉也可以帮助判断多至三维的数据聚类分析质量。然而设计对高维空间中的数据对象,特别是对高维空间稀疏和怪异分布的数据对象,能进行较好聚类分析的聚类算法已成为聚类研究中的一项挑战。

(8) 基于约束的聚类。现实世界中的应用可能需要在各种约束之下进行聚类分析。假设需要在一个城市中确定一些新加油站的位置,就需要考虑诸如城市中的河流、高速路,以及每个区域的客户需求等约束情况下居民住地的聚类分析。设计能够发现满足特定约束条件且具有较好聚类质量的聚类算法也是一个重要的聚类研究任务。

(9) 可解释性和可用性。用户往往希望聚类结果是可理解的、可解释的,以及可用的。这就需要聚类分析与特定的解释和应用联系在一起。因此研究一个应用的目标是如何影响聚类方法的选择也是非常重要的。

5.4 基于 K-means 的文本聚类算法

5.4.1 概述

目前的文本聚类方法大致可以分为层次凝聚法和平面划分法两种类型。层次聚类又被称作系统聚类,通过将已有的类别两两比较,找出最相近的类别合并,最终所有的数据都被聚到单一的类别中。由于有较大的搜索空间,能够生成层次化的嵌套簇,层次聚类比

平面划分法容易获得较高的精度；但是，在每次合并时，需要全局地比较所有簇之间的相似度，并选择出最佳的两个簇，因此运行速度较慢，不适合于大量文档的集合。平面划分法与层次凝聚法的区别在于，它将文档集合水平地分割为若干簇，而不是生成层次化的嵌套簇。平面划分方法可以取得较好的运算速度。但由于文本聚类本身没有机器学习过程，在事先不知道类别的情况下对文本进行自动匹配和归类，因而具有盲目性。对于平面划分方法而言，一般需要在初始时对一些对聚类效果有决定性作用的参数进行设置，因而这些参数的合理选择就显得至关重要。聚类算法中初始化时常涉及的参数有聚类个数、相似度阈值、允许的迭代最大次数、类内分散程度的参数等。

层次凝聚法的代表如 HAC 算法，平面划分法的代表如 K-means 算法。其中，K-means 文本聚类算法理论上可靠、算法简单、速度快且易实现。一般地，K-means 文本聚类算法以 k 为参数，把 n 个文档对象分为 k 个簇。K-means 算法聚类是一种无监督的学习。无监督学习的目标是在非标记训练数据中发现隐藏的结构和模式，是可以将相同群组或者聚类的成员，在某种衡量标准下相互之间比和其他聚类的成员更相似。在利用该算法进行聚类操作时，k 的选取对结果和过程均有着不同程度的影响，其中，n 是文档集合中所有文档的数目，k 是簇的数目。

5.4.2 K-means 算法理论基础

K-means 算法中，需要不断计算向量距离并进行迭代操作。在数学方法中有三种较为常见的方法：欧氏距离(Euclidean Distance)、曼哈顿距离(Manhattan Distance)和余弦夹角(Cosine)。

欧氏距离是数学中表示向量距离的最常用的计算方法，在二维空间表示的是两个向量连线的直线长度。二维空间欧氏距离的数学公式为：

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

其中，(x_1,y_1)和(x_2,y_2)是空间中的两个点。

若是多维空间各个点之间的绝对距离，公式如下：

$$d(X,Y)=\sqrt{\sum_{i=1}^{n}(x_i-y_i)^2}$$

在进行文本聚类分析时，多采用多维空间欧氏距离公式。欧氏距离是用 K-means 算法求距离最常用的算法。在下文中所使用的 K-means 算法均采用欧式距离进行计算。

当对一组数据开始进行聚类并选取了聚类中心后，开始迭代过程。K-means 算法将待分配实例分配到距离最近的聚类中，然后将图心移动到观测值的均值位置。K-means 算法参数的最优值是通过最小化一个代价函数来决定的。代价函数的公式如下：

$$J=\sum_{k=1}^{K}\sum_{i\in C_k}\|x_i-\mu_k\|^2$$

在此处 μ_k 表示聚类 k 的中心，这个代价函数是对所有聚类的偏差求和。每个聚类的偏差等于其包含的所有实例和其图心之间距离的平方和，此处需要采用上文的欧氏距离进行代入计算。

每次迭代后，聚类中心均可能会有所变化，则需要再次进行计算和重新分配待聚类实例到合适位置。K-means 算法会一直进行迭代直到满足某种标准。通常情况下，这个标准是当前代价函数值和后续迭代代价函数值之间的差值的阈值，或者是当前图心位置和后续迭代图心位置变化的阈值。如果这些停止标准足够小，K-means 将会收敛到一个最优值。然而，随着停止标准值的减小，收敛所需的时间会增大。

5.4.3 K-means 算法结果影响因素

要生成的簇数目 k 的选择，是影响聚类结果的一个重要因素。如果 k 值太大，聚类则会过细；如果 k 值过小，测试样本太多，类的中心主题词不能全面覆盖该类的内容。k 值选取的依据可以是主观判断，用户或有经验的专家以各自领域知识判断，以经验和反复验证为基础。另外，k 值的选取也可以依据数学论形式进行客观判断，如衡量 k 取不同值时所对应的聚类效果优劣的紧致与分离性效果函数以及与误差平方和函数相结合的标准 JW 准则。

另外，初始聚类中心点不同，聚类结果也会不同。一般常用的初始聚类中心点选取的方法有：任意随机地选取 k 个样本作为初始聚类中心；凭经验选取有代表性的点作为初始聚类中心，根据个体性质，观察数据结构，选出比较合适的代表点；把全部混合样本直观地分成 k 类，计算各类均值作为初始聚类中心；进行多次初值选择、聚类，找出一组最优的聚类结果；等等。

5.4.4 TF-IDF 理论基础

TF-IDF(Term Frequency-Inverse Document Frequency，词频-逆向文件频率)是一种用于信息检索与文本挖掘的常用加权技术。

TF-IDF 是一种统计方法，用以评估一字词对于一个文件集或一个语料库中的其中一份文件的重要程度。字词的重要性随着它在文件中出现的次数成正比增加，但同时会随着它在语料库中出现的频率成反比下降。其主要思想是：如果某个单词在一篇文章中出现的频率 TF 高，并且在其他文章中很少出现，则认为此词或者短语具有很好的类别区分能力，适合用来分类。

词频(TF)表示词条(关键字)在文本中出现的频率。这个数字通常会被归一化，以防止它偏向长的文件。

$$\mathrm{tf}_{ij}=\frac{n_{i,j}}{\sum_k n_{k,j}} \quad \text{即 } \mathrm{TF}_w=\frac{\text{在某一类中词条 } w \text{ 出现的次数}}{\text{该类中所有的词条数目}}$$

其中，$n_{i,j}$ 是该词在文件 d_j 中出现的次数，分母是文件 d_j 中所有词汇出现的次数总和。

逆向文件频率 (IDF)：某一特定词语的 IDF，可以由总文件数目除以包含该词语的文件的数目，再将得到的商取对数得到。如果包含词条 t 的文档越少，IDF 越大，则说明词条具有很好的类别区分能力。

$$\mathrm{idf}_i=\lg\frac{|D|}{|\{j:t_i\in d_j\}|} \quad \text{即 } \mathrm{IDF}=\lg\left(\frac{\text{语料库的文档总数}}{\text{包含词条 } w \text{ 的文档数}+1}\right)$$

其中，$|D|$是语料库中的文件总数；$|\{j:t_i\in d_j\}|$表示包含词语t_i的文件数目（即$n_{i,j}\neq 0$的文件数目）。如果该词语不在语料库中，就会导致分母为零，因此一般情况下使用$1+|\{j:t_i\in d_j\}|$。

结合TF和IDF的定义以及公式，某一特定文件内的高词语频率，以及该词语在整个文件集合中的低文件频率，可以产生出高权重的TF-IDF。因此，TF-IDF倾向于过滤掉常见的词语，保留重要的词语。TF-IDF实际上是：TF－IDF＝TF×IDF。

例如，总数据集有10 000篇文章，其中一篇文档总共有1000个词，其中，“文本聚类”出现了5次，“的”出现了25次，“应用”出现了12次，那么它们的词频（TF）分别是0.005、0.025和0.012。且“文本聚类”只在其中10篇文章中出现，则其权重指数为$\mathrm{IDF}=\lg\left(\frac{10\,000}{10+1}\right)=3$。“的”在所有文章中均出现过，则其权重指数为$\mathrm{IDF}=\lg\left(\frac{10\,000}{10\,000+1}\right)=0$。“应用”在其中1000篇文章中出现过，则其权重指数为$\mathrm{IDF}=\lg\left(\frac{10\,000}{1000+1}\right)=1$。所以可以得到这三个词语的TF-IDF分别为0.015，0和0.012。也就是说，“文本聚类”的重要性在这三个词中是最高的。

5.4.5 基于K-means文本聚类的主要步骤

首先随机地选择k个初始文本对象，每个对象代表了一个簇的平均值或中心。对剩余的每个对象，根据其与各个簇中心的距离，将它赋给最近的簇。然后重新计算每个簇的平均值。这个过程不断重复，直到准则函数收敛，或中心趋于稳定为止。下面给出了基于K-means的文本聚类算法的形式化描述。算法的复杂度是$O(nkt)$，其中，n是文档集合中所有文档的数目，k是簇的数目，t是迭代的次数。

给定文档集合$D=\{d_1,\cdots,d_i,\cdots,d_n\}$，K-means文本积累算法具体过程如下。

（1）确定要生成簇的数目k。

（2）按某种原则选取k个初始聚类中心$C=(c_1,c_2,\cdots,c_k)$，并设置初始迭代次数$r=1$。

（3）对文档集中的每一个文档d_i，依次计算它与各个聚类中心c_j的相似度$\mathrm{sim}(d_i,c_j)$。

（4）选择具有最大相似度的聚类中心$\mathrm{argmaxsim}(d_i,c_j)$，将$d_i$归入以$c_j$为中心的簇中。

（5）计算新的聚类中心。新的聚类中心为这一轮迭代中分到该簇中的所有文档矢量的均值，即$c_j=\frac{1}{n_j}\sum_{d\in F_j}d$，其中，$F_j$为聚簇$c_j$的文档集合，$n_j$为$F_j$中的文档数。

（6）如果所有聚类中心均达到文档，则结束；否则，$r=r+1$，转到（3）。

5.4.6 基于K-means算法的聚类实例

K-means算法可以用于数据集的聚类，采用一个常用的二维数据集——4k2_far作为测试样本，如表5-2所示。

表 5-2　测试样本集的部分样例

x_1	x_2
7.1741	5.2429
6.914	5.0772
7.5856	5.3146
6.7756	5.1347
…	…

其中，x_1，x_2 表示数据集中样本的属性。随着不断迭代，质心也不断地接近每个簇的中心位置。并且根据不同的 k 的取值，最终的聚类结果也会有所不同。

从图 5-1 可以看出，当选取了不同的 k 值时，一个数据集聚类的结果会有所不同。但也并非是 k 越大越好，可以对比看出在 k 取 4 时对于整体的划分是较为合适的，过少会导致类别划分不够明显，过多会导致在计算过程中浪费时间，有些类别划分也过于刻意。

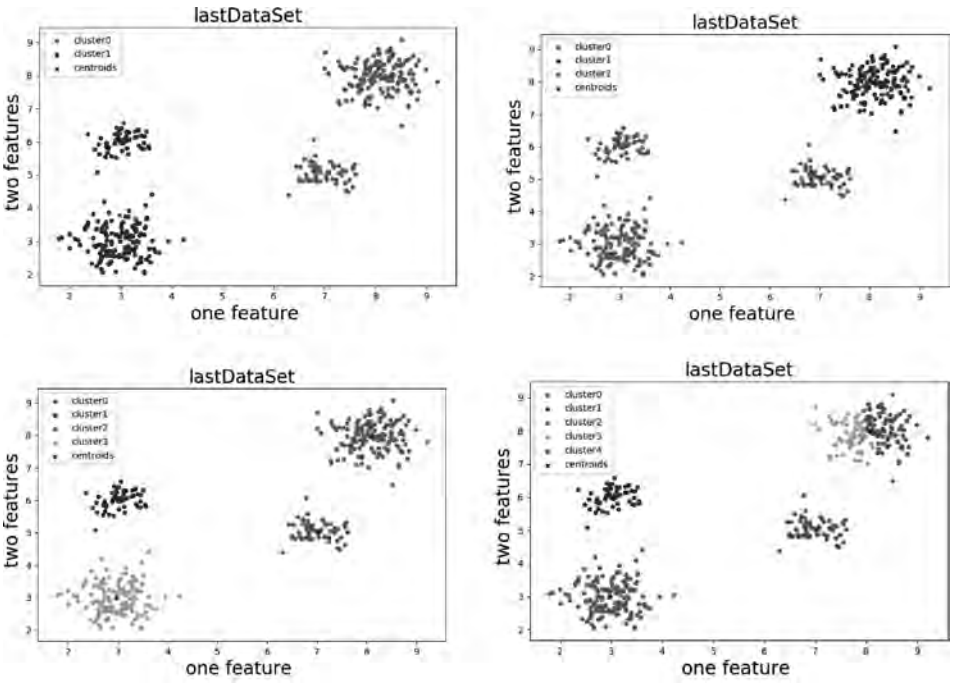

图 5-1　当 k 取 2、3、4、5 时的不同聚类结果

当使用 K-means 算法进行文本聚类时，一般会对文本数据集进行预处理，例如，利用 jieba 分词进行文本分词、停用词过滤、语料库的建立等，利用 scikit-learn 计算词语的 TF-IDF。在对文本进行了上述预处理后，方可使用 K-means 算法进行文本聚类。

可以采用来自 mlcomp.org 上的 20news-18828 数据集进行 K-means 算法的文本聚类。在该数据集中，train 子目录下有 20 个子目录，每个子目录代表一种文档的类型，为了进一步简化问题，只选择语料库里的部分内容来进行聚类分析。假设选择 sci.crypt、sci.electronics、sci.med 和 sci.space 共 4 个类别的文档进行聚类分析。首先将待聚类的

文本先进行文本预处理，进行分词并计算各个词的 TF-IDF 值，设置 max_df 和 min_df 以确保词频过高和过低的词语影响分类结果。通过以上对文本的预处理后，可以使用 K-means 算法进行文本聚类。

设置聚类个数为 4 个，K-means 迭代最多进行 100 次，当中心点移动距离小于 0.1 时默认为算法已经达到收敛。在每次进行迭代和聚类的过程中，均是使用各个关键词的重要权重(TD-IDF)进行欧氏距离计算并比较。通过 3 次 K-means 聚类分析，分别做了 195 424 次迭代后可以实现收敛。与此同时将 3949 个文档进行自动分类。可以通过查询，查询到对应的文档被分到了哪一类中，也可以查询到对应的文件名。选取部分 K-means 算法聚类后的结果与实际情况进行对比，如表 5-3 所示。

表 5-3 部分 K-means 算法文本聚类结果与实际情况对比

文本号	K-means 算法聚类结果	文本实际所在类别	聚类是否正确
1000	1	sci. crypt	正确
1001	1	sci. crypt	正确
1002	1	sci. crypt	正确
1003	0	sci. electronics	正确
1004	3	sci. space	正确
1005	1	sci. crypt	正确
1006	2	sci. electronics	错误

通过结果可以看出文本相对应的分类基本上被正确聚类。但是在该数据集中，第一个分类 sci. electronics 的特征词比较普遍，没有过于有特征的特点，会导致其聚类结果不是很好。但其余几组因为较高权重词语的指向性较强，聚类结果也相对较好。

5.5 基于潜在语义索引的文本聚类方法

5.5.1 概述

为了克服传统 VSM 模型的局限性，S. T. Dumains 等人提出了一种新的模型：潜在语义索引，或者潜在语义分析，本书取其英文简写 LSI。LSI 可以看作一种扩展的向量空间模型，它利用统计计算导出文本中隐含的语义，而不是表面上的词的匹配。LSI 基于这样的一种断言，即文本库中存在隐含的关于词使用的语义结构，这种结构由于部分地被文本中词的语义和形式上的多样性所掩盖而不明显。LSI 通过对原文档中词-文档矩阵的奇异值分解计算，并取前 k 个最大的奇异值对应的奇异向量构成一个新的矩阵来近似地表示原文本库的词-文本矩阵，再对此矩阵进行相关的文本处理操作，这就是 LSI 技术。下面将分别讨论 LSI 的相关理论。

5.5.2 矩阵的奇异值分解

在论述矩阵的奇异值与奇异值分解之前，先看下面的结论和定理。

(1) 设 $\boldsymbol{A}\in C_r^{m\times n}(r>0)$，则 $\boldsymbol{A}^H\boldsymbol{A}$ 是 Hermite 矩阵，且其特征值均是非负数。

(2) $\mathrm{rank}(\boldsymbol{A}^H\boldsymbol{A})=\mathrm{rank}(\boldsymbol{A})$。

(3) 设 $\boldsymbol{A}\in C_r^{m\times n}(r>0)$,则 $\boldsymbol{A}=0$ 的充要条件是 $\boldsymbol{A}^H\boldsymbol{A}=0$。

定义 5.2.1 设 $\boldsymbol{A}\in C_r^{m\times n}(r>0)$,则 $\boldsymbol{A}^H\boldsymbol{A}$ 的特征值为:

$$\lambda_1\geqslant\lambda_2\geqslant\cdots\geqslant\lambda_r\geqslant\lambda_{r+1}=\cdots=\lambda_n=0$$

则称 $\sigma_i=\sqrt{\lambda_i}(i=1,2,\cdots,n)$为 $\boldsymbol{A}$ 的奇异值。

定理 5.1 设 $\boldsymbol{A}\in C_r^{m\times n}(r>0)$,则存在 m 阶矩阵 $\boldsymbol{U}$ 和 n 阶矩阵 $\boldsymbol{V}$,使得

$$\boldsymbol{U}^H\boldsymbol{A}\boldsymbol{V}=\begin{bmatrix}\sum & 0\\ 0 & 0\end{bmatrix} \tag{5-1}$$

其中,$\sum=\mathrm{diag}(\sigma_1,\sigma_2,\cdots,\sigma_r)$,而 $\sigma_i(i=1,2,\cdots,r)$为矩阵 $\boldsymbol{A}$ 的全部非零奇异值。

式(5-1)可变换为:

$$\boldsymbol{A}=\boldsymbol{U}\begin{bmatrix}\sum & 0\\ 0 & 0\end{bmatrix}\boldsymbol{V}^H \tag{5-2}$$

称式(5-2)为矩阵的奇异值分解。

如矩阵 $\boldsymbol{A}=\begin{bmatrix}1 & 0 & 1\\ 0 & 1 & 1\\ 0 & 0 & 0\end{bmatrix}$,则 $\boldsymbol{A}$ 的奇异值分解为:

$$\boldsymbol{A}=\boldsymbol{U}\begin{bmatrix}\sum & 0\\ 0 & 0\end{bmatrix}\boldsymbol{V}^H \tag{5-3}$$

其中,

$$\sum=\begin{pmatrix}\sqrt{3} & 0\\ 0 & 0\end{pmatrix}\quad \boldsymbol{U}=\begin{pmatrix}\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0\\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0\\ 0 & 0 & 1\end{pmatrix}\quad \boldsymbol{V}=\begin{pmatrix}\frac{1}{\sqrt{6}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{3}}\\ \frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{3}}\\ \frac{2}{\sqrt{6}} & 0 & -\frac{1}{\sqrt{3}}\end{pmatrix}$$

5.5.3 LSI 技术的理论基础

1. 词-文档矩阵

LSI 矩阵模型中,一个文档库可以表示为一个 $m\times n$ 的词-文档矩阵(Term-Document)$\boldsymbol{A}$。这里 n 表示文本库中的文本数;m 表示本库中包含的所有不同的词的个数。这样,每个不同的词对应于矩阵 $\boldsymbol{A}$ 的一行,而每个文本则对应于矩阵的一列。$\boldsymbol{A}$ 可表示为:

$$\boldsymbol{A}=[a_{ij}] \tag{5-4}$$

其中,a_{ij} 为非负值,表示第 i 个词在第 j 个文本中出现的频度。由于词和文档的数量很大,而单个文本的词的数量又非常有限,所以 $\boldsymbol{A}$ 一般为稀疏矩阵。

2. 权重的选取

通常 a_{ij} 要考虑来自两个方面的贡献，即局部权值 $L(i,j)$ 和全局权值 $C(i)$，它们分别表示第 i 个词在第 j 个文本和在整个文本库中的重要程度，有：

$$a_{ij}=L(i,j)\times C(i) \tag{5-5}$$

局部权值 $L(i,j)$ 和全局权值 $C(i)$ 有不同的取值方法。

表 5-4 和表 5-5 列出了局部权值计算方法和全局权值计算方法。其中，tf_{ij} 和 Gf_i 分别表示词 i 在文本 j 和整个文本库中出现的频度；df_i 为文本库中包含词 i 的文本数目；ndocs 为文本库中文本的总数，即 $p_{ij}=\mathrm{tf}_{ij}/\mathrm{Gf}_i$。

表 5-4 局部权值计算方法

方 法 名	公 式	备 注
词频法	tf_{ij}	
0/1 二值法	0/1	词在文档中存在时为 1，否则为 0
对数词频法	$\lg(\mathrm{tf}_{ij}+1)$	

表 5-5 全局权值的计算方法

方 法 名	公 式
Normal	$\sqrt{1/\sum_j \mathrm{tf}_{ij}^2}$
Gfidf	$\mathrm{Gf}_i\,\mathrm{df}_i$
Idf	$\lg\left(\frac{\mathrm{ndocs}}{\mathrm{df}_i}\right)+1$
Entropy	$1-\sum_j \frac{p_{ij}\lg(p_{ij})}{\lg(\mathrm{ndocs})}$

5.5.4 基于 LSI 文本聚类的主要步骤

基于 LSI 文本聚类的主要步骤如下。

(1) 对文本库中的文本进行切词处理，构建词-文档矩阵 $\boldsymbol{A}$，并计算 $\boldsymbol{A}$ 中各个元素的权值。

(2) 对 $\boldsymbol{A}$ 进行奇异值分解，得到 $\boldsymbol{V}_k$，文本库中的所有文本对应 $\boldsymbol{V}_k$ 中的一行。

(3) 利用某种向量间的相似性度量，依据某种聚类算法计算 $\boldsymbol{V}_k$ 的行向量(每个文本对应一条行向量)之间的相似度进行聚类。

文档 i,j 之间的相似度可利用 $\boldsymbol{V}_k$ 的对应行向量之间的相似度来求得，计算公式为：

$$\cos(\theta_{ij})=\frac{(e_l\boldsymbol{V}_k)(e_j\boldsymbol{V}_k)^{\mathrm{T}}}{\|e_j\boldsymbol{V}_k\|_2\,\|(e_j\boldsymbol{V}_k)^{\mathrm{T}}\|_2} \tag{5-6}$$

其中，e_i 表示 k 阶 6 单位矩阵的第 i 列。

5.5.5 基于 LSI 文本聚类的实例

下面将展示一个较为简单的利用 LSI 算法进行文本聚类的例子。当在 Amazon.com

上搜索“investing”时将返回 10 个书名，这些书名都有共同的一个索引词。一个索引词可以是符合以下条件的任何单词。

（1）出现在两个或以上的文章题目中。

（2）停止词：词义过于一般，如“and”“the”等。这些词对文章的语义并没起到突出的作用，因此应该被过滤掉，也就是当作“停用词”。

以下展示的是索引出来的 9 个标题，粗体字为索引词。

（1）The Neatest Little **Guide** to **Stock Market Investing**
（2）**Investing** ** **For** ** **Dummies**，4th Edition
（3）The Little **Book** ** **of Common Sense** ** **nvesting**：The Only Way to Guarantee Your Fair Share of **Stock Market** Returns
（4）The Little **Book** of **Value Investing**
（5）**Value Investing**：From Graham to Buffett and Beyond
（6）**Rich Dad's Guide** to **Investing**：What the **Rich** Invest in，That the Poor and the Middle Class Do Not!
（7）**Investing** in **Real Estate**，5th Edition
（8）**Stock Investing** For **Dummies**
（9）**Rich Dad's** Advisors：The ABC's of **Real Estate Investing**：The Secrets of Finding Hidden Profits Most Investors Miss

首先，LSI 需要创建单词-标题矩阵。在该矩阵中，行表示索引词，而列表示题目。每个元素表示对应的标题包含多少个相应的索引词。例如，“book” 在 T3 和 T4 中出现了 1 次，而“investing” 出现在所有的表中。一般情况下，LSI 创建的单词-标题矩阵会相对巨大，而且十分稀疏（大部分元素为 0），这是因为每个标题或文章一般只包含十分少的频繁单词。改进的 LSI 通过这种稀疏性能有效降低内存的损耗和算法复杂度。构建的单词-标题矩阵如表 5-6 所示。

表 5-6 LSI 构建的单词-标题矩阵

索引词	标　题								
	T1	T2	T3	T4	T5	T6	T7	T8	T9
book			1	1					
dads						1			1
dummies		1						1	
estate							1		1
guide	1					1			
investing	1	1	1	1	1	1	1	1	1
market	1		1						
real							1		1
rich						2			1
stock	1		1					1	
value				1	1				

在 LSI 算法中，源单词-标题（或文章）矩阵一般会进行加权调整，其中稀少的词的权

重会大于一般性的单词。因为这个例子规模不大，因此不对矩阵进行权重调整。

当单词-标题（或文章）矩阵创建完成，将使用强大的 SVD 算法进行矩阵分析。为了确定合适的有效维度，通过奇异值的平方的直方图来进行观察。图 5-2 中演示出各奇异值的重要性。

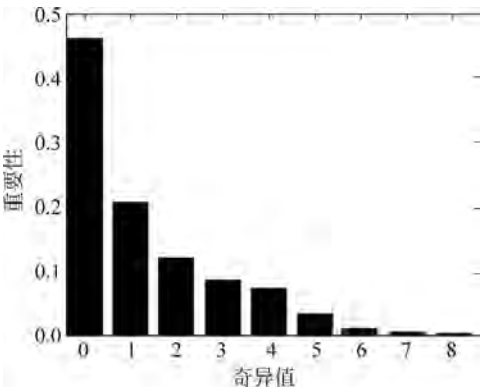

图 5-2　奇异值结果展示

为了实现可视化，选择有效维度数为 3。最后，将选择第 2 维和第 3 维进行可视化。我们不对单词-标题（或文章）矩阵进行中心化，是为了避免将单词-标题（或文章）矩阵由稀疏矩阵变为稠密矩阵。稠密矩阵会增加内存的负荷和计算量。因此不对单词-标题（或文章）矩阵进行中心化和放弃第 1 维的做法更加高效。

这里计算出了 3 个奇异值，分别对应着 3 个维度。每个单词的这 3 个维度与这些奇异值相关，第 1 维表示该单词在语料库中的频繁程度，因此没有太大信息量。类似地，每篇文章也有 3 个维度分别对着 3 个奇异值。如之前所述，第 1 维反映了文章所包含索引词的数量，因此信息不大。将矩阵分解成 3 个矩阵。矩阵 $\boldsymbol{U}$ 提供了每个单词在语义空间的坐标；矩阵 $\boldsymbol{V}^{\mathrm{T}}$ 提供了每篇文章在语义空间的坐标；奇异值矩阵 $\boldsymbol{S}$ 告诉我们有词-标题（或文章）矩阵包含多少语义或语义空间的有效维度是多少。

在如图 5-3 所示的三个矩阵中，左奇异向量表示词的一些特性，右奇异向量表示文档的一些特性，中间的奇异值矩阵表示左奇异向量的一行与右奇异向量的一列的重要程序，数字越大越重要。除此之外，左奇异向量的第一列表示每一个词的出现频繁程度，虽然不是线性的，但是可以认为是一个大概的描述，例如，book 是 0.15 对应文档中出现的 2 次，investing 是 0.74 对应文档中出现了 9 次，rich 是 0.36 对应文档中出现了 3 次。另外，右奇异向量中的第一行表示每一篇文档中出现词索引的个数的量化，例如，T6 是 0.49，出现了 5 个索引词，T2 是 0.22，出现了 2 个索引词。

book	0.15	-0.27	0.04
dads	0.24	0.38	-0.09
dumales	0.13	-0.17	0.07
estate	0.18	0.19	0.45
guide	0.22	0.09	-0.46
investing	0.74	-0.21	0.21
market	0.18	-0.30	-0.28
real	0.18	0.19	0.45
rich	0.36	0.59	-0.34
stock	0.25	-0.42	-0.28
value	0.12	-0.14	0.23

*

3.91	0	0
0	2.61	0
0	0	2.00

*

T1	T2	T3	T4	T5	T6	T7	T8	T9
0.35	0.22	0.34	0.26	0.22	0.49	0.28	0.39	0.44
-0.32	-0.15	-0.46	-0.24	-0.14	0.55	0.07	-0.33	0.44
-0.41	0.14	-0.16	0.25	0.22	-0.51	0.55	0.00	0.34

图 5-3　三个矩阵

我们用不同的颜色表示数字。如图 5-4 所示，用颜色来表示 $\boldsymbol{V}^{\mathrm{T}}$ 矩阵的值，这个颜色表示的矩阵和原 $\boldsymbol{V}^{\mathrm{T}}$ 矩阵反映的信息完全一致。深灰色表示负数，浅灰色表示正数，白色表示 0。如标题 9，其 3 个维度上的值都是正数，因此相应的颜色都是浅灰色。

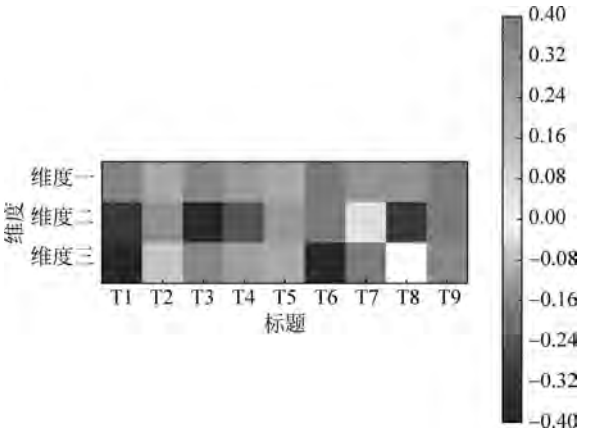

图 5-4　颜色直观表示各标题聚类情况

用这些颜色对聚类结果进行颜色标注。忽略第 1 维表示的颜色，因为所有文章在该维度上都是橙色。如果只考虑两个维度，则聚类的结果仍然不是很理想，主要也只分为两类，所以需要结合三个维度进行聚类。加上第 3 维，我们能用相同的方法区分出不同的语义群。在第 3 维上，标题 6 是蓝色，而标题 7 和标题 9 依然是橙色的。通过这种方法将标题集分成 4 个群，如表 5-7 所示。

表 5-7　标题集群

维度 2	维度 3	标题序号
橙色	橙色	7,9
橙色	蓝色	6
蓝色	橙色	2,4,5,8
蓝色	蓝色	1,3

最后，将矩阵 $\boldsymbol{U}$ 和 $\boldsymbol{V}$ 的第 2,3 维画在一个二维 XY 平面中，其中，X 表示第 2 维，Y 表示第 3 维，并将所有索引词和标题画在该平面中。如图 5-5 所示，单词“book”的坐标值为 (0.15，−0.27，0.04)，忽略第 1 维的值 0.15 后，“book” 的坐标点为 ($X=-0.27$，$Y=0.04$)。标题的画法也是类似的，如图 5-5 所示。

通过可视化方法可以将单词和标题都画在同一个空间。这种做法不仅能实现标题的聚类，还能通过索引词标注出不同类簇的意义。例如，左下的簇包含标题 1 和标题 2，这两个标题均关于 stock market investing。单词“stock”和“market”明显包含在标题 1 和标题 2 的簇中，这也很容易理解这个语义簇所指代的意义。中间的簇包含标题 2,4,5,8。其中，标题 2,4,5 与单词“value” 和“investing”代表的意思最为接近，因此，标题 2,4,5 的语义可表示为“value” 和“investing”。

按这样聚类出现的效果，可以提取文档集合中的近义词，这样当用户检索文档的时

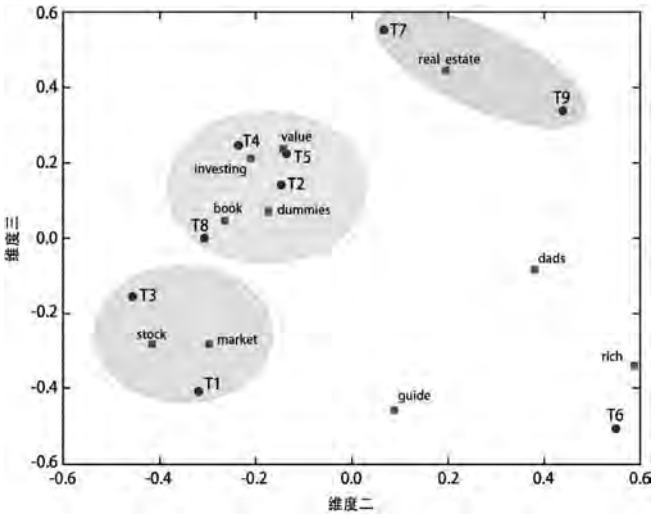

图 5-5 聚类结果展示

候，是用语义级别（近义词集合）去检索了，而不是之前的词的级别。这样做一是减少了检索、存储量，因为这样压缩的文档集合和 PCA 是异曲同工的；二是可以提高用户体验，用户输入一个词，我们可以在这个词的近义词的集合中去找，这是传统的索引无法做到的。

5.6 基于 Word2Vec 的文本聚类方法

5.6.1 词向量概述

计算机想要通过机器学习算法处理自然语言，就需要将自然语言符号化、数学化、转换成机器能识别的格式，其中，词向量就是目前被广泛使用的方式。词向量是由 Hinton 和 Williams 等提出并推广的，现在词向量已经被广泛使用在各种文本挖掘任务中，极大地促进了 NLP 领域的发展。词向量有两种表示形式，一个是长向量（又称为稀疏向量），一个是短向量（又称为密集向量）。

稀疏向量，又称为独热编码向量。独热编码，顾名思义就是指向量中只有一个热点，热点的位置就是此向量表达的含义，并且向量之间是相互独立的。独热编码向量中只有 0 和 1，1 对应的位置就是特征词在语料词典中的位置。对于[“水果”，“香蕉”，“手机”]这个词典来说，若“水果”的词向量是[1，0，0]，那么“香蕉”对应的词向量就是[0，1，0]。这种向量表示方法很容易实现，对于语料库比较小的数据集来说，很有学习意义，但是对于大数据集，这种方法会造成向量的维数灾难，并且计算复杂，性能低下；其次，对于近义词它无法区别出来。如水果和香蕉显然有非常紧密的关系，但转换为向量之后，就看不出两者之间的关系了，因为这两个向量相互正交。

密集向量也就是分布式向量（Distributed Representation），相当于把原来的独热编码向量压缩成一个长度更短的向量，向量中的数值不再只有 0 和 1，而是任意数字。分布式向量通过输入语料中每个词的独热编码，根据特征词的上下文环境，将每个词的编码向

量训练成具有相同长度的低维实数向量，这种方法很好地表达了近义词之间的关系。还是之前的例子["水果","香蕉","手机''],假设经过训练后,"水果"对应的向量可能是[1,0,1,1,0],而"香蕉"对应的向量可能是[1,0,1,0,0],"手机"对应的向量可能是[0,1,0,0,1]。这样"水果"向量乘以"香蕉"向量＝2,而"水果"向量乘以"手机"向量＝0。这样就很明显看到水果和香蕉有很紧密的联系,而水果和手机就没有什么关系了。

5.6.2　Word2Vec 语言模型

Word2Vec 是 Mikolov 在 2013 年提出的将特征词转换为词向量的模型。模型根据特征词的上下文预测特征的词向量,由于词向量用低维实数表示,看起来无意义的向量却蕴含丰富的信息,它保持了同义词之间强的相关性,并且很好地根据特征词推测其所在的上下文环境。图 5-6 展现了 Word2Vec 的算法模型。

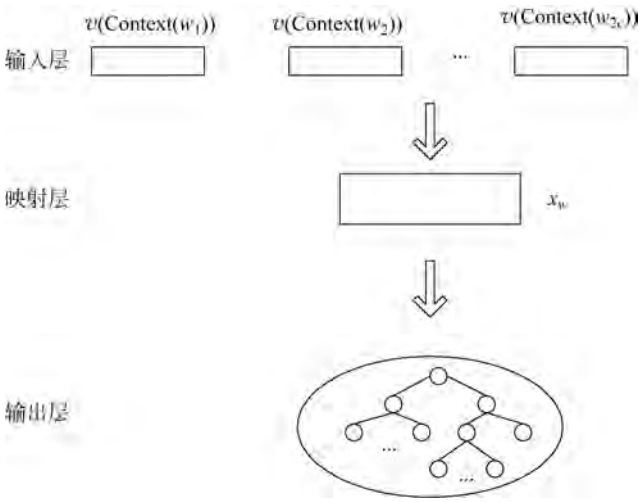

图 5-6　Word2Vec 模型

CBOW 模型和 Skip-gram 模型均包含输入层、投影层和输出层。CBOW 模型的输入是特征词的上下文环境,其中每个词的输入是词对应的独热编码,经过模型计算映射成低维的实数向量,之后通过变换矩阵,输出预测词的独热编码向量。而 Skip-gram 模型输入的是特征词的独热编码,经过矩阵变换之后,输出的是特征词周围可能出现词的独热编码,独热编码中 1 的位置指示了词典中对应的词。其中,Context(w)表示特征词的上下文,由前后 c 个词构成；投影层是对输入层的每个词对应的独热编码向量进行简单求和,其中有变换矩阵；输出层对应一棵 Huffman 树,该树以每个词在语料中出现的权值构造出来。

5.6.3　连续词袋模型

连续词袋模型(Continuous Bag-Of-Wordmodel,CBOW)是通过某个特征词的上下文环境来预测这个特征词。假设有一个句子结构为 $w_{i-2}w_{i-1}w_iw_{i+1}w_{i+2}$,CBOW 就是通过输入 $w_{i-2}w_{i-1}w_{i+1}w_{i+2}$ 的词向量,来预测 w 的词向量。其结构如图 5-7 所示。

其中,V 表示特征词词典的大小,C 表示窗口的大小。$\{x_{1k},x_{2k},\cdots,x_{ck}\}$ 表示待预测

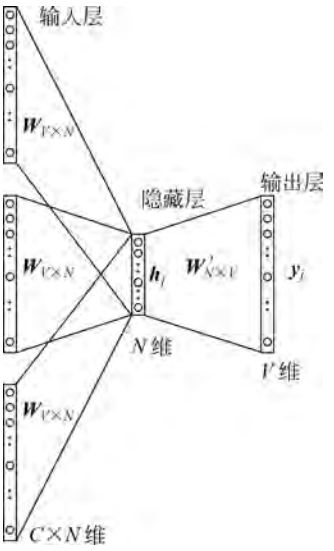

图 5-7 CBOW 结构

的特征词上下文环境词语的独热编码向量。对于每一个输入的向量，经过矩阵 $\boldsymbol{W}_{V\times N}$ 的变换后对应到隐含层的一个分量上。隐含层是一个 N 维的向量 $\boldsymbol{h}$，输出层是预测特征词 y 的独热编码的向量。CBOW 的公式如下。

$$P(W_i \mid W_{i-k},\cdots,W_{i-1},W_{i+1},\cdots,W_{i+k}) \tag{5-7}$$

W_i 表示一个单词，$W_{i-k},\cdots,W_{i-1},W_{i+1},\cdots,W_{i+k}$ 是其邻居，根据其邻居的独热编码向量来预测它自己出现的概率。在预测的过程中，从 Huffman 树的根结点出发到某个叶子结点的路径上，通过二分类方法来决定路径是往左分支走还是往右分支走。

右分支：

$$\sigma(X_\omega^{\mathrm{T}}\theta)=\frac{1}{1+\mathrm{e}^{-X_\omega^{\mathrm{T}}\theta}} \tag{5-8}$$

左分支：

$$1-\sigma(X_\omega^{\mathrm{T}}\theta) \tag{5-9}$$

公式中，θ 代表当前非叶结点的词向量。

对于 Huffman 树中的任意一条路径 P^w，存在 l^w-1 次分支，将每次分支看成一个二分类，每次分类对应一个概率，那最后预测特征词的概率将是这些概率连乘，即

$$p(w \mid \mathrm{Context}(w))=\prod_{j=1}^{l^w} p(d_j^w \mid X_w,\theta_{j-1}^w) \tag{5-10}$$

其中：

$$p(d_j^w \mid X_w,\theta_{j-1}^w)=\begin{cases}\sigma(X_\omega^{\mathrm{T}}\theta_{j-1}) & d_j^w=0\\ 1-\sigma(X_\omega^{\mathrm{T}}\theta_{j-1}) & d_j^w=0\end{cases} \tag{5-11}$$

5.6.4 Skip-gram 模型

Skip-gram 模型是通过一个特定的特征词来预测这个词的周围邻居可能出现的词，如果将预测窗口定为 k，则它的预测大小为 $2k-l$。假设这个特定词窗口的大小为 C，则输出层为 w_i 的上下文 $\{w_0,w_1,\cdots,w_{2k}\}$。例如，考虑这个句子“I drove my car to the store”，一个潜在的训练模型就是将“car”作为输入，其他词作为输出，这些词都是以 one-hot 向量编码格式，无论是输入还是输出，向量的长度是字典的大小 V。Skip-gram 模型的神经网络如图 5-8 所示。

图 5-8 中，x 代表输入，指的是特征词的独热编码向量。$\{y_1,y_2,\cdots,y_c\}$ 也是独热编码格式的向量，作为模型的输出。y_i 中元素为 1 的位置表

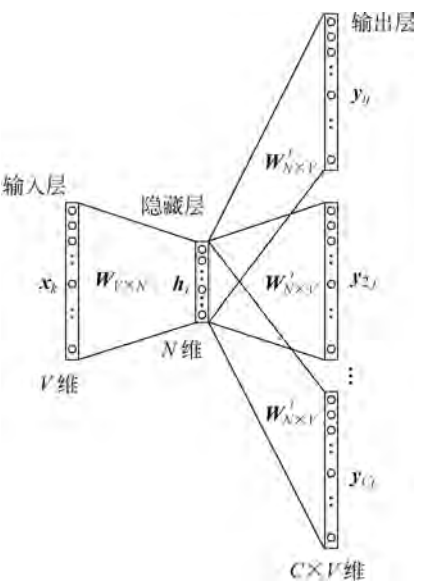

图 5-8 Skip-gram 模型

示该位置有词语，元素为 0 表示该位置没有词语。矩阵 $\boldsymbol{W}$ 是一个 $V\times N$ 的权重矩阵，连接着输入层和隐含层，它包含词典中全部词语的向量编码信息。每个输出的词向量都关联着 $\boldsymbol{W}\times\boldsymbol{V}$ 的权重矩阵 $\boldsymbol{W}'$。Skip-gram 的公式如下：

$$p(w_i \mid w_t),\quad t-k\leqslant i\leqslant t+k \tag{5-12}$$

对于 Skip-gram 模型来说，输出层也是一棵 Huffman 树，也是通过二分类方法预测其特征词的上下文。公式如下：

$$p(\text{Context}(w)\backslash w)=\prod_{u\in\text{Context}(w)} p(u \mid w) \tag{5-13}$$

其中，$p(u \mid w)=\sum_{j=2}^{l^w} p(d_j^u \mid v(w),\theta_{j-1}^u)$。

5.6.5 基于 Word2Vec 的文本聚类举例

Word2Vec 工具的提出很好地解决了独热词向量两两正交而无法准确表达不同词之间的相似度且会产生一个维度很高又十分稀疏的特征矩阵而难以应用实际的问题，故而可以利用 Word2Vec 进行文本表示，进而结合 TF-IDF 用于进行无监督的短文本的文本聚类。

简而言之，可以利用 TF-IDF 方法提取短文本中的 TOP N 关键词，作为短文本的特征词集合，有效避免特征词向量维度过高、数据稀疏及计算效率低效等问题。另一方面，可以使用 Word2Vec 将特征词表示为一种分布式的低维实数向量，使得语义相近的词语在距离上更加接近，有效解决了单独使用 TF-IDF 方法存在的语义丢失问题。

基于 Word2Vec 短文本的文本聚类流程如图 5-9 所示。

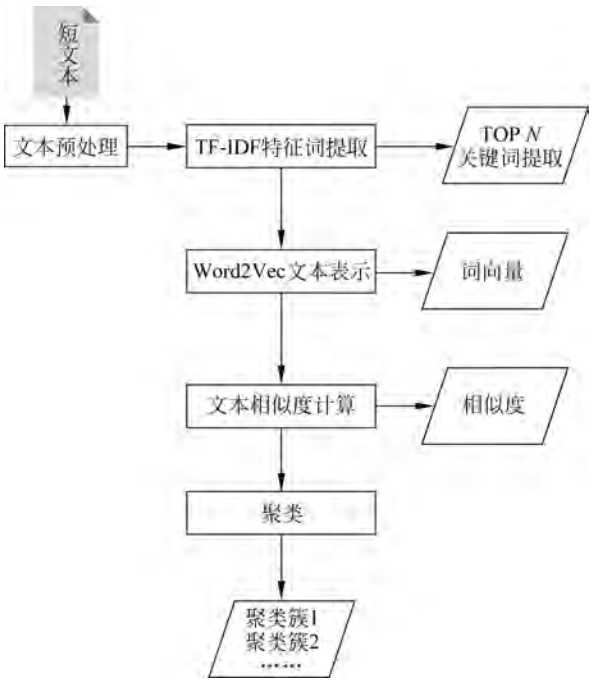

图 5-9 基于 Word2Vec 短文本的文本聚类流程图

根据如上分析，可以总结基于 Word2Vec 文本聚类的主要步骤如下。

1. 文本预处理

在对文本分词后、关键词提取之前，需要对文本进行预处理。具体处理步骤如下：采用停用词列表，过滤掉文本中对应于停用词列表中的词项。

2. 利用 TF-IDF 进行特征词提取

特征词是指提取能够代替短文本特征的词语。具体而言，针对每个短文本，首先计算该文本中各分词的 TF-IDF 值；其次，为尽可能减少文本特征向量的维度，把每一个文本中计算得到的各分词的 TF-IDF 值进行排序，从中选取 TF-IDF 值 TOP N 靠前的词项作为文本的特征词集合，用于表示该文本，其中，N 为百分比。

3. 利用 Word2Vec 得到文本库中每个词语的词向量表示

由于 Skip-gram 模型在语义和语法预测的准确率方面比较均衡且性能要好，因此使用 Word2Vec 的 Skip-gram 模型，通过 HS-Huffman 对爬取的网络短文本语料进行词向量模型训练，根据当前输入层的词项，预测上下文词项出现的概率，并且选择时间窗口为 2。

通过训练得到词向量模型可以得到特征词的向量，组成特征词向量矩阵 $\boldsymbol{X} \in R^{mT}$，其中，$m$ 为特征词 i 在 m 维向量空间的向量，T 表示特征词的数目。由此，设两个特征词的特征向量分别为 $x_i, x_j \in \boldsymbol{X}$，则两个特征词之间的相似度可以用欧氏距离来计算，值越小，说明这两个特征词间的语义距离越小，两个特征词语义越相似。

4. 文本相似度计算

利用 Word2Vec 得到短文本的特征向量以后，接下来就可以进行文本相似度计算，为文本聚类做准备。实际上，计算文本的相似度，已经被转换为计算特征词向量间的相似度。下面采取的距离函数是 Kusner 等提出的 WMD(Word Mover's Distance)。WMD 具有效果出色、无监督、模型简单、可解释性、灵活性等优点，并且充分利用了 Word2Vec 的领域迁移能力。

WMD 的核心思想非常简单，可以把特征文本的相似度计算看成一个运输问题：计算将仓库 1 的货物移动到仓库 2 的最小距离，作为两个文本的相似度，而所谓“仓库”中的“货物”指的是文本中的特征词。

图 5-10 是 WMD 的一个应用举例图，通过 WMD 算法可以得出，文档 1 中的 Obama、speaks、Illinois 和 media 分别与文档 2 中的 President、greets、Chicago 和 press 语义相近。

WMD 算法用以上核心思想将文本语义相似度的问题转换成了一个线性规划问题，为最优化问题。

由 Word2Vec 训练后得到的特征词向量矩阵 $\boldsymbol{X} \in R^{mT}$，其中索引值为 i 的词 x_i 和相应的索引值为 j 的词 x_j 的距离为欧氏距离。

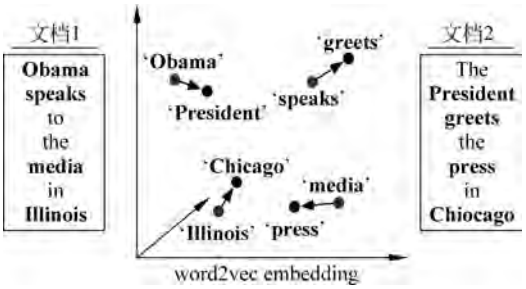

图 5-10 WMD 的应用举例图

$$c(i,j)=\|x_i-x_j\|^2 \tag{5-14}$$

而待分类的某条文本可以用一个稀疏向量 $\boldsymbol{d}\in R^n$ 作为词袋表示，若该文本中词 i 的出现频率为 c_i，则 $\boldsymbol{d}$ 的第 i 位是第 i 个词的词频 $\boldsymbol{d}_i$，即

$$\boldsymbol{d}_i=\frac{c_i}{\sum_{j=1}^{n}c_j} \tag{5-15}$$

故引入词移距离(WMD)即

$$\sum_{i,j=1}^{n}T_{ij}c(i,j) \tag{5-16}$$

其中，T_{ij} 表示某两个文本中特征词 x_i 移动到 x_j 的数值，需要满足以下约束条件：

$$\sum_{i,j=1}^{n}T_{ij}=\boldsymbol{d}_i\ \forall i\in\{1,2,\cdots,n\} \tag{5-17}$$

$$\sum_{i,j=1}^{n}T_{ij}=\boldsymbol{d}_j\ \forall j\in\{1,2,\cdots,n\} \tag{5-18}$$

最后求解其最小值即可。

$$\min_{T\geqslant 0}\sum_{i,j=1}^{n}T_{ij}c(i,j) \tag{5-19}$$

5. K-means 聚类

使用 K-means 算法对文本集进行聚类分析，输入欲分类的短文本集，以期得到 k 个类簇集合。

经过前述步骤得到基于特征词向量的短文本集合 $D=\{d'_1,d'_2,\cdots,d'_n\}$，设定 k 的具体值，从 D 中随机选择 k 个短文本作为聚类算法的初始质心；计算每个短文本 d'_j 到 k 个质心的文本相似度，选择最短聚类的质心作为该文本的簇集合。重新计算类簇中所有短文本的距离平均值得到新的质心，取到质心最近的文本作为新的质心；循环以上几步直到质心不再发生任何变化，输出 k 个类簇集合。

本次实验选取的数据集是 20-newsgroup、3Cphys 两种英文数据集和通过网络爬取的中文数据集即微博数据集和微信聊天数据集，如表 5-8 所示。

表 5-8 数据集描述

数 据 集	类 别 数	数 量	长 度
20-newsgroup	5	349	6.71
3Cphys	3	1066	9.39
微博数据集	5	18 771	8.25
微信聊天数据集	7	21 356	8.51

在实验中，根据 TD-IDF 本文选择 TOP N 的值为 35%。对于两种英文数据集，特征向量维度设置为 100，最低词个数设置为 3，窗口大小设置为 3；对于两种中文数据集，特征向量维度设置为 270，最低词个数设置为 4，窗口大小设置为 4。

为评价短文本的聚类效果，引入 F 度量作为评价指标。短文本的聚类效果是否良好，取决于每个文本聚类后能否正确分类以及分类后的每个类别下是否都包含所有正确分类的短文本。因此定义查准率 $P(i,j)$和查全率 $R(i,j)$：

$$P(i,j)=\frac{m_{ij}}{m_j} \tag{5-20}$$

$$R(i,j)=\frac{m_{ij}}{m_i} \tag{5-21}$$

其中，m_i 是类别 i 的文本数量，m_j 是聚类 j 的文本数量，m_{ij} 是聚类 j 中属于类别 i 的文本数目。

对应的 F 度量值 $F(i,j)$定义为：

$$F(i,j)=\frac{2\times P(i,j)\times R(i,j)}{P(i,j)+R(i,j)} \tag{5-22}$$

全局聚类的 F 度量值为：

$$F=\sum_i \frac{m_i}{m}\max_j(F(i,j)) \tag{5-23}$$

其中，m 是数据集中文本的总数量。故 F 度量值越大，聚类效果越好，文本相似度度量也越好。

实验结果如表 5-9 所示，F 值越大说明方法的聚类效果越好。

表 5-9 不同数据集聚类后的 F 值

数 据 集	F 值
20-newsgroup	0.456
3Cphys	0.493
微博数据集	0.376
微信聊天数据集	0.412

习题

1. 常用聚类方法有哪些？它们各自的特点是什么？

2. 聚类技术和分类技术的区别是什么？

3. 聚类算法的评价指标是什么？

4. 基于 K-means 的文本聚类算法属于常用聚类方法中的哪一类？简述该算法步骤。

5. 简述基于 Word2Vec 的文本聚类算法。

6. 现有 5 个二维数据样本组成的数据对象集 S，如表 5-10 所示。

表 5-10 数据对象集 *S*

ID	X	Y
1	0	2
2	0	0
3	1.5	0
4	5	0
5	5	2

要求簇的数目为 $K=2$，试用 K-means 算法对集合 S 进行聚类。

7. 表 5-11 为亚洲 15 支球队在 2005—2010 年间大型杯赛战绩（由于澳大利亚是后来加入亚足联的，所以这里没有收录），试选用任意一种分类算法分析中国男足在 2005—2010 年处于亚洲足球的什么水平？

表 5-11 亚洲球队杯赛战绩

	2006 年世界杯	2010 年世界杯	2007 年亚洲杯
中国	50	50	9
日本	28	9	4
韩国	17	15	3
伊朗	25	40	5
沙特	28	40	2
伊拉克	50	50	1
卡塔尔	50	40	9
阿联酋	50	40	9
乌兹别克斯坦	40	40	5
泰国	50	50	9
越南	50	50	5
阿曼	50	50	9
巴林	40	40	9
朝鲜	40	32	17
印尼	50	50	9