分类号：TP391.1　　单位代码：10058

天津工業大學
TIANGONG UNIVERSITY

工程硕士学位论文

论文题目： 基于中国在线食谱的探索式数据分析

工程领域： 计算机技术

学习方式： ☑全日制攻读□非全日制攻读

作者姓名： 刘兆沛

学校导师： 荣垂田

企业导师： 李颖

完成日期： 2019-11-30

独 创 性 声 明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作和取得的研究成果，除了文中特别加以标注和致谢之处外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得 天津工业大学 或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

学位论文作者签名：　　签字日期：　　年　　月　　日

学位论文版权使用授权书

本学位论文作者完全了解 天津工业大学 有关保留、使用学位论文的规定。特授权 天津工业大学 可以将学位论文的全部或部分内容编入有关数据库进行检索，并采用影印、缩印或扫描等复制手段保存、汇编以供查阅和借阅。同意学校向国家有关部门或机构送交论文的复印件和磁盘。

（保密的学位论文在解密后适用本授权说明）

学位论文作者签名：　　导师签名：

签字日期：　年　月　日　　签字日期：　年　月　日

摘 要

随着中国在线食谱网站及应用的不断发展，越来越多来自不同地区的用户将他们制作的食谱分享到线上，为我们研究中国在线食谱提供了丰富的数据资源。中国作为一个地域广阔，人口众多的国家，孕育着丰富的饮食文化，衍生了多种菜系，其中最为知名的有中国八大菜系。目前基于数据驱动的中国食谱和烹饪（饮食）习惯研究还较少，我们通过数据采集技术爬取到了大量的在线食谱数据，并以这些食谱数据作为研究基础，对中国各大菜系展开探索式地数据分析。

在本文中，我们对中国在线食谱从如下几个方面进行了探索。首先，本文对在线食谱的成分多样性展开分析，包括成分消费多样性及成分组合多样性；接着对中国各个菜系食谱的特色成分进行探索，并以词云的形式将结果进行可视化；其次，我们对中国在线食谱的复杂性从成分数量、烹饪时间、烹饪工艺三个方面进行了评估；再次，根据食谱的成分、口味、烹饪工艺这些特征，我们构造了食谱的特征向量空间，从而能通过文本相似度算法对食谱之间的特征向量进行相似性的分析，并通过可视化的技术进一步地探索它们所在菜系之间的关联性；最后，我们对在线食谱的辅料成分进行频繁项的挖掘，以探索经常使用的辅料成分组合。

另外，互联网上对于标签的使用越来越频繁，一些社交软件如微博、推特上的许多内容都被赋予了多种标签，以方便话题搜索。而对于在线食谱而言，它们的附加标签表示的是食谱的膳食功能，例如“补钙”和“抗氧化”等。因此，对于多标签自动分类的业务需求也相应越来越广泛，而传统机器学习模型处理分类任务时通常需要繁琐的特征工程，并且它们大部分都是二分类模型，对于多标签分类任务的处理往往需要进行模型的调整和转化。为了取得更好的多标签分类效果，我们通过在深度学习平台上搭建卷积神经网络、循环神经网络，完成对在线食谱的多标签分类任务，并与传统机器学习分类模型进行性能对比。同时，为了进一步扩充在线食谱数据的训练样本，我们采用了文本数据增强技术，以期望获得更好的多标签分类效果。

关键词：在线食谱；数据分析；数据挖掘；深度学习；多标签分类

ABSTRACT

With the rapid development of food-focused websites and applications in China, more and more cooking recipes are published on the websites by the people from different regions, which provides us rich data resources for the study of online recipes in China. As a big country with a large population, China has generated a series of different cuisines. The most famous of them are eight major cuisines. However, lack of research focus on Chinese recipes from the aspect of data analysis. We adopt data crawler technology to extract enough recipe data, and use these recipe data as our research object to analysis the Chinese cuisine.

In this paper, we explore Chinese online recipes from the following aspects. Firstly, we explore the diversity of ingredients, including the diversity of ingredient consumption and combination. Then we explore the notable ingredients of recipes between various Chinese cuisines, and the results are visualized in the form of word cloud. Secondly, we explore the complexity of Chinese online recipes from three aspects: the number of ingredients, cooking time and cooking way. Thirdly, for the recipe feature vectors composed of ingredients, taste and cooking way, we use similarity algorithm to analyze the similarity between them. In this way we can use data visualization technology to explore the relationship between the cuisines in which these recipes belonged to. Finally, we try to find the frequent items for minor ingredients of Chinese online recipes, such that we can obtain the frequently combinations of minor ingredients for Chinese cuisines.

In addition, labels are widely used on many social networks and applications, like Weibo and Twitter. Many contents within them have been given a variety of labels in order to facilitate topic search. For online recipes, their labels indicate the dietary functions, such as “calcium supplementation”, “antioxidant” et al. Therefore, the requirements for multi-label automatic classification are becoming more and more extensive. However, the traditional machine learning methods usually need complex feature engineering to deal with the classification tasks, and most of them are binary classification methods that needed to be transformed into multi-label classification methods. In order to achieve better performance of multi-label classification, we complete the multi-label classification task of Chinese recipes by using convolution neural network and recurrent neural network on the deep learning platform, and also compare their performance with the machine learning classification methods. At the same time, we try to expand the training sets of Chinese recipe data by using text data augmentation technology for better performance.

Keywords: Online recipe; Data analysis; Data mining; Deep learning; Multi-label classification

目 录

第一章 绪论

1.1 本文的研究背景及意义

食物是一个基本而重要的话题[1]，它影响着我们生活和文化的方方面面。在几千年的发展过程中，中国已经在不同的地区培育了不同的菜系和独特的饮食文化。以往关于中国饮食习惯的研究，受限于数据采集技术的落后及高质量数据集的匮乏，不能很好地得以开展。近年来随着社交应用的兴起，人们热衷于在线上分享各地的美食，而专门以食谱制作与分享为主题的在线社区也应运而生。在线美食社区的出现，聚集了全国各地爱好美食制作的人们，他们是社区发展的主力军。在经过多年的发展和积累，中国境内的食谱网站建设规模越来越大。以美食杰在线食谱网站为代表，截止至目前，该网站拥有国内领先数量的视频菜谱及图文菜谱近百万篇，每年超过 1 亿次的浏览量，目前为止超过 5000 万家庭享用该网站提供的饮食指南。如此丰富的线上资源为本文开展关于中国在线食谱的研究提供了数据来源，摆脱了以往研究缺乏食谱数据资源的困境，并为最新研究提供了更具有说服力的数据依据。

不同的菜系是不同文化、地区和习俗的重要体现。中国是一个历史悠久的大国，幅员辽阔，人口众多。来自中国不同地区和民族的人们开发出了自己独特的地方菜品。中国菜系大致可划分为四川、湖南、广东、福建、浙江、江苏、山东、安徽等地的八大菜系。其他一些菜系也有一定知名度，比如北京菜和上海菜。基于在线食谱网站的数据，如何分析菜系的成分多样性，描绘菜系之间的联系性，挖掘每个菜系食谱特有的成分，以及定量地衡量出各菜系食谱的复杂性等，是本文前半部分研究的核心所在。通过上述的分析，将有助于推动我们对中国在线食谱相关特性的了解，丰富对中国饮食文化在数据层面上的认知。

于此同时，标签功能被广泛地应用在当今的许多社交应用上，如微博、领英。用户的任何一条微博都可以加入相应标签，这些标签可以代表相关内容的主题或特征。对于在线食谱网站而言，标签指示的是食谱的食疗功能，如“补钙”标签出现在“麻婆豆腐”这道食谱上。标签的应用不仅能帮助用户快速地找到所需的内容，还可帮助网站开发者整合主题相近的内容并增强用户个性化推荐服务。然而，只有少数食谱包含标签由作者创建。因此，如何根据在线发布的内容自动分类出相应的标签，变得尤为重要。

综上，在线食谱网站的发展为我们研究中国在线食谱提供了丰富的数据源，

通过对食谱数据展开多种维度地数据分析与挖掘，将有助于我们从数据层面去窥探中国各菜系食谱的一些特征和内在联系。同时，自动标签分类功能的实现将有利于用户的使用以及在线食谱网站的推广。

1.2 国内外的研究现状

因缺乏高质量的数据集，从而限制了以往关于在线食谱的研究。伴随着以食谱分享为主题的社交应用的发展，在线食谱的数据量迅猛增加，这一变化打开了对于在线食谱的研究窗口。针对在线食谱的研究，有包括对于在线食谱消费和生产模式的研究、对食谱成分营养价值的研究等。

同时，互联网对于内容标签的使用十分频繁，它方便了用户准确地检索到想要了解的内容。对于在线食谱中存在的标签，它们通常代表了相应的膳食功能，这对于爱好美食的用户也是非常重要的一部分，并且食谱标签体系的建立对于网站的管理和维护也大有裨益。目前围绕着在线食谱分析和多标签文本分类有诸多相关的研究，下面我们将分别从这两个部分对国内外研究现状进行阐述。

1.2.1 在线食谱相关研究

在线食谱消费和生产模式的研究是一个相对较新的研究领域，到目前为止只有少数相关研究。虽然这一领域的传统研究大多是通过例如在线或问卷调查[2]进行的，但目前已使用了更先进的研究方法，即将统计和数据挖掘技术应用在诸如在线食谱社区或社交媒体等平台，以便更大规模地展开相关研究。在这方面，第一个使用大规模食谱数据集的研究是由 Ahn 等人[3]进行的，他们挖掘并分析了一个拥有庞大数据量的在线食谱社区平台 allrecipes.com，以揭示食谱是如何在全球范围内从口味角度上创造的，同时还发现了各个国家地区的食谱在成分制作上的显著差异。Teng 等人[4]随后紧跟着进行了一项有趣的研究，同样基于 allrecipes.com 平台上的食谱数据，他们采用与 Ahn 等人类似的方法，训练出了一个能够向用户推荐食谱的统计模型。

也有一些研究探索了人们在线食谱消费的模式，在这方面最突出的工作是 West 等人[5]进行的一项研究。在他们的工作中，分析了用户访问食谱的日志文件。除此以外，他们还发现美国不同地区的人们对营养价值偏好的季节性趋势。同时，他们还发现在美国的某些地区，在线食谱消费模式与心脏病之间存在关联。一项有趣的后续研究是，Wagner 等人[6]在欧洲的一个在线食品社区平台上调查了在线食谱消费的动态，并且得到的结果与 West 等人的研究结果类似。他们还发现在线食谱的消费方式与人们所在的地区存在明显的时间模式，强调了美国和欧洲在

线社区之间的显著差异。与前面提到工作类似的是 Said 和 Bellogín[7]的研究，他们的工作同样基于在线食谱社区 allrecipes.com，他们发现美国肥胖症病例与在线食谱消费及生产的显著相关性；Abbar 等人基于 Twitter 建立了一个预测肥胖和糖尿病的统计数据模型[8]，他们研究也取得了类似的结果。最后，Kusmierczyk 等人从在线食谱成分消费和创造的角度观察到了营养价值偏好的差异[9]。

围绕食谱也有很多应用型的研究，如 Müller 等人设计了一个计算食谱成分营养价值的系统[10]。在他们的研究中，首先将食谱的成分与标准营养表进行匹配，然后基于它们去推断营养事实。De Choudhury 等人[11]也采取了类似的做法，他们在研究中使用了美国农业部国家营养数据库，并从 Instagram 关于食谱的内容中获取成分的营养和热量等信息。Min W 等人[12]提出了一个能够综合利用食材、食谱图片以及多种属性的饮食文化分析框架，实现了多种应用。他们设计的 BC^2TM 模型能够结合烹饪和食谱信息来发现相关主题，模型采用了多种排序方法检索与主题相关的食谱图片，并实现主题可视化。最后他们将主题建模和可视化方法应用于多模式烹饪总结、烹饪-食谱模式分析和食谱推荐三个方面，并揭示了饮食文化的地区多样性。Nedovic V[13]利用了 LDA 和 DBN 算法来学习食谱成分分布的生成模型，并得到了一些新的食谱成分组合。

对于中国在线食谱，Zhu Y X 等人[14]研究了中国主要区域的菜系在气候和地理位置上的关系，他们通过控制地理距离因素，发现了气候(温度)与食谱成分使用的相似性没有任何关联性，而地理上的接近似乎是形成区域菜系的一个关键因素。

1.2.2 多标签分类相关研究

多标签分类是自然语言中文本分类的重要部分。对于多标签文本，它允许数个标签被赋予在一个样本数据上，并且标签之间的关系通常是层次结构，它可以从多个维度描述样本的特征属性。多标签分类算法的实现可以分为监督学习和无监督学习，同时对于多标签分类算法的处理通常有两种方式，一是进行问题转化，例如将多标签分类转化为多个二分类问题，最后构成分类器链进行预测得到最终分类结果，二是直接改编算法以适应多标签分类任务。

对于第一种解决方式，Read, J 等人[15]提出了一种多标签分类的链式分类方法。该方法通过沿着分类器链传递标签相关信息，克服了二进制关联方法的缺点，在保持较低计算复杂度的同时，获得了较高的预测性能。并且使用分类器链的集合来消除无序链的可能性，从而进一步提高预测性能。与专门的改编型分类算法不同，分类器链的集合可以被看作是一种通用的“现成”方法，不需要参数配置，并且在一系列数据中具有健壮的性能。除了实现高性能之外，它们还具有很高的

可伸缩性，并且能够在我们考虑的最大数据集上完成。Tsoumakas G 等人提出了 RAkEL 方法[16]，该方法学习一组 LP 分类器，每个分类器对应一组标签中不同的小随机子集。同时该方法还考虑了标准 LP 方法在具有大量标签和训练实例情况下的计算效率和预测性能问题。

对于第二种解决方式，K. Dembczynski 等人提出了基于概率分类器链的贝叶斯最优多标签分类[17]。他们的研究扩展了最近引入的 CC 分类器。该算法首先能够估计标签的整个联合分布，其次可以根据不同的损失函数进行适应，最后以最优的方式将复杂性与概率估计的准确性进行权衡。同时，他们还研究了多标签分类中的标签依赖问题[18]。Ji, S.等人提出了一种在多标签分类中提取共享结构的通用框架[19]。在这个框架中，多个标签之间的相关信息被所有标签之间共享的低维子空间捕获。他们的研究表明，当在分类中使用最小二乘损失函数时，可以通过求解广义特征值来计算共享结构。Xia, X.等人提出了一种新的多标签分类算法 RW.KNN[20]。RW.KNN 结合了 KNN 和 Radom Walk 模型的优点，为多标签学习提供了一个新的视角。他们研究的基于 KNN 的连接图使得多标记数据集中的随机游动成为可能，同时还提出了一种新的基于最小 Hamming 损失评价指标的分类阈值计算算法。

一些研究工作从标签本身出发进行了探索，如 Zhang M L 提出了一种基于标签特征多标签学习策略，即一种简单而有效的 Lift 算法[21]。Lift 算法构造了每个标签的特性，先对其正实例和负实例进行聚类分析，然后通过查询聚类结果来进行训练和测试。该研究已在 16 个不同的数据集上进行了广泛的实验，很好地验证了 Lift 相对于其他已建立的多标签学习算法的优越性。同时 M.-L.等人还利用标签之间的依赖关系[22]，采用贝叶斯网络来表示标签空间在特征空间上的联合分布，能够对任意阶的标签相关性进行建模。他们还找到了一种有效的方法来找到近似的网络，即通过处理所有标签的分类错误，而不是原始标签。研究最后还说明了学习系统的复杂性与可能的标签数成线性关系。

同时，Wang H 等人研究了多标签分类中一对多方法所产生的训练不一致问题，利用非参数分类方法得到了一种多标签分类的平衡 K 近邻（BKNN）方法[23]，解决了非参数分类方法中的问题，强调了所有类的数据的均衡使用。最后还提出了一种将高维多标签输入数据嵌入到较低维空间中的类平衡线性判别分析(BLDA)方法，该方法可用于分类前的可选降维。

由于多标签样本中的标签是呈多层次结构的，而对于处理多标签文本分类问题的两种处理方式而言，它们共同的缺点就是忽视了多个标签之间的联系，将对分类器的预测造成误差。而以词嵌入(word embedding)[24]作为输入的深度学习模型在多标签文本分类的任务中显示出了优良的效果，其中多标签文本分类中的深

度学习模型包括 fastText[25]、卷积神经网络(CNN)[26]、循环神经网络(RNN)[27]以及 CNN 与 RNN 的组合 RCNN[28]等。

1.3 本文的主要研究内容

本文针对中国在线食谱展开多种维度的数据分析，主要维度有食谱的成分、口味、烹饪方法及时间，旨在从数据层面上客观地分析中国食谱的重要特征以及内在联系；并应用可视化的技术对分析出来的结果进行直观展示。同时，爬虫框架的应用帮助了我们采集到了重要的数据资源，解决了以往研究食谱难以获取数据资源的瓶颈。最后，为了解决人工对文本内容添加标签的繁琐，我们针对多标签分类任务搭建了卷积神经网络模型，并采用词向量技术对食谱成分进行了预训练；对于输入的训练数据采用了数据增强的处理，同时与传统机器学习分类器进行了对比。本文主要研究内容主要分为两个部分：

（1）在线食谱的数据采集与分析：在数据采集部分，设计并实现了基于 Python 的 Scrapy 爬虫框架，该框架从食谱网站中提取到了大量的食谱数据，并对数据进行了数据清洗工作。在数据分析部分，首先通过计算食谱成分分布的信息熵，来衡量不同菜系食谱的多样性，包括成分消费多样性及成分组合多样性；接着通过对各菜系食谱的成分文档的 TF-IDF 计算，得到权值最高的 K 个成分视为特殊的成分，并进行可视化；其次，通过从食谱的成分、口味、烹饪工艺、烹饪时间等方面综合对各菜系食谱的复杂性进行分析；再次，将所有食谱进行向量化表示，对这些向量使用余弦相似进行计算得到它们的相似值，其中食谱向量的元素包括食谱的成分、口味及烹饪工艺；最后，通过频繁项挖掘算法，找出所有菜系中频繁使用的辅料成分集。

（2）基于食谱成分的多标签分类任务：在 Pytorch 深度学习平台下，通过搭建卷积神经网络(CNN)和循环神经网络(RNN)，在经过 word2vec 预训练后的食谱成分上进行训练，并进行大量的实验以优化参数设置，得出我们认为性能最好的多标签分类器；实验表明，数据增强的使用，可以丰富我们训练的样本，提升分类器的准确性。最后，通过将神经网络模型与传统机器学习模型进行性能对比，突出地体现了前者在多标签分类任务中的优势。

1.4 本文的组织结构

本文共由五个章节构成，各章节核心内容如下：

第一章为绪论部分，主要介绍了本文的研究背景及意义、国内外的研究现状、

本文的主要研究内容以及本文的组织结构。

第二章为研究基础和相关技术部分，主要包括了数据采集技术、文本挖掘相关技术、特征工程与文本分类和深度学习技术。

第三章为在线食谱数据的采集与分析部分，主要包括在线食谱数据采集模块、在线食谱的成分的多样性分析、在线食谱的特色成分分析、在线食谱的复杂性分析、在线食谱的相似性分析以及在线食谱的惯用辅料分析。

第四章为基于食谱成分的多标签文本分类部分，主要包括了传统机器学习算法的实现、深度学习分类模型的实现、实验环境与结果分析以及小结。

第五章为本文研究工作的总结与展望，对所得出的研究成果进行概括性的总结，并提出今后所要研究的方向。

第二章 研究基础和相关技术

本章节主要介绍本文所用到的相关技术。本章共分为五部分，第一部分介绍数据采集技术；第二部分介绍文本挖掘相关技术；第三部分介绍特征工程与文本分类；第四部分介绍深度学习技术；第五部分为本章小结。

2.1 数据采集技术

2.1.1 网络爬虫技术

网络爬虫[29]是一种程序脚本或软件，也被称为网络蜘蛛或软件代理，它是搜索引擎的主要组成部分。Internet 是一个有向图，其中网页作为节点，超链接作为有向边，搜索网页的操作可以概括为遍历有向图的过程。通过设定规则使爬虫遵循 Web 的链接结构之后，网络爬虫能够以自动化的方式浏览网络中的信息，这些规则我们称之为网络爬虫算法。网络爬虫由爬虫节点、控制节点、资源库三部分组成，图 2-1 表示了网络爬虫的爬虫节点和控制节点之间的结构关系。

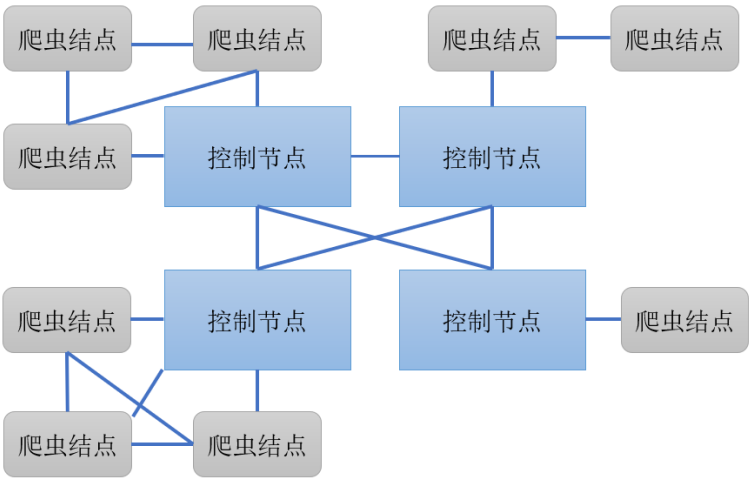

图 2-1 网络爬虫的各节点结构关系

根据网络爬虫实现的技术及结构可将它们细分为通用网络爬虫、聚焦网络爬虫、分布式式网络爬虫、深层网络爬虫等多种类型。在实际的使用场景中，通常将这几类爬虫的进行组合使用。

其中通用网络爬虫主要应用于搜索引擎中，它的特点是爬行范围非常宽广、数据量庞大、对网络带宽的消耗较大，因此这类爬虫对性能要求非常高。通用爬虫在爬行的时候会采取深度优先爬行和广度优先爬行这两种策略。

聚焦网络爬虫是只用于收集特定主题的文档，它不像通用网络爬虫一样拥有宽广的爬行范围，而是将爬行的目标聚焦在与主题相关的页面中；此时，可以节省相当一部分的网络带宽和服务器资源。聚焦网络爬虫应用在预先定义的一组事项中，主要服务于某一类特定的人群。分布式爬虫能够使用多个进程来完成工作，能够更好地发挥机器的性能。

深度网络爬虫能够对互联网中的深层页面展开爬取，而深度网络爬虫最重要的部分即为表单填写部分。因为深层页面无法直接通过静态链接获取，需要人工设计一些方法自动填写好相应的表单才能够获取相应的页面。深度网络爬虫有两种表单填写类型：第一种是基于领域知识的表单填写，即预先建立好包含领域相关知识的关键词库，在对表单进行填写时，能够结合语义的分析来选择相关性较强的关键词来完成填写；第二种是基于网页结构分析的表单填写，这种方式通常在领域知识有限的情况下使用，该方式将依据网页结构完成分析，并进行表单自动填写。

网络爬虫的工作从初始 URL 开始，它们下载网页内容并提取已下载页面中出现的新链接，同时存储检索到的网页并在存储区域建立相应的索引，以便用户之后能够更好地在这些区域检索它们。对于前面已下载好的页面中提取到的新 URL，需要被确认是否已下载或未下载；如果未下载这些 URL，则再次将 URL 分配给网络爬虫进行进一步的下载，重复此过程，直到找不到更多的新 URL。本文在图 2-2 说明了爬行过程，具体由以下步骤：

(1) 选择一个初始 URL 或 URLs；

(2) 将该 URL 添加到待完成列表 frontier；

(3) 从 frontier 中选取 URL；

(4) 爬虫对相应 URL 的网页内容进行下载；

(5) 解析网页内容并提取新的 URL 链接；

(6) 将新发现的 URLs 添加至 frontier；

(7) 跳转回步骤 3，直到 frontier 为空。

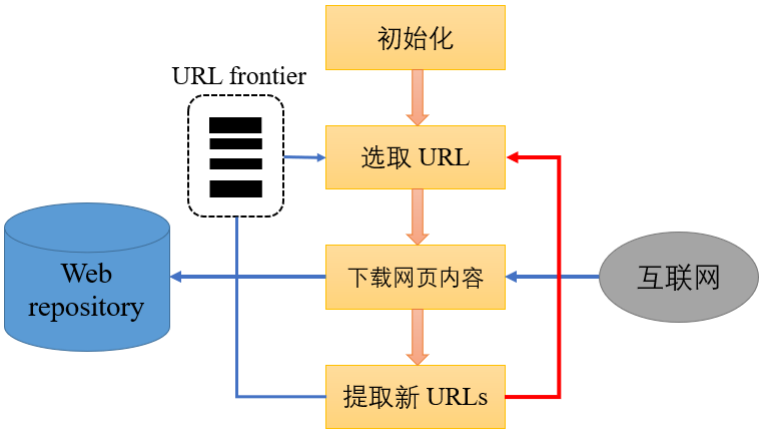

图 2-2 网络爬虫的工作流程

2.1.2 Scrapy 爬虫框架

Scrapy[30]是基于 Python 网络框架 Twisted 开发的一个高效率信息抓取框架，能够快速而有效地抓取 Web 页面并提取出结构化的数据，常广泛应用于数据采集、数据挖掘和数据监控等场景。它主要由引擎(Engine)、调度器(Scheduler)、下载器(Downloader)、爬虫(Spider)、管道(Pipeline)、下载中间件(Downloader Middlewares)、爬虫中间件等组件构成，具体如图 2-3 所示。

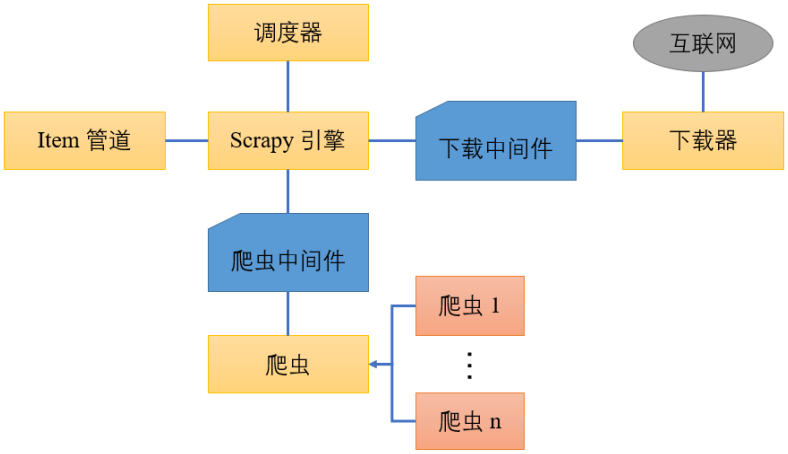

图 2-3 Scrapy 框架组成

其中，引擎(Engine)是框架的核心，负责调控系统所有组件之间的数据流，并在相应动作发生时触发事件。调度器(Scheduler)主要接收来自引擎的请求，并在引擎请求时为它们提供下一次要抓取的网址，以供引擎进行进一步的处理。可以将调度器的存储结构当作一个优先队列，调度器将从引擎中接受到的 request

请求存入优先队列中，队列中可能存在多个待抓取的网址，但这些网址都具备一定的优先级，并且调度器会过滤掉重复的网址，以避免重复爬取。

下载器(Downloader)先通过引擎传过来的 Requests 请求对要抓取的网页内容进行高速下载，再将获取到的数据传递给引擎，最后由引擎传给相应的爬虫进行处理。该组件需要进行大量的数据传输，消耗更多的网络带宽，因此该组件的负荷通常比其他组件更重。

爬虫(Spider)是整个框架实现的核心，它是我们编写的自定义类。在一个 Scrapy 项目中，通常存在多个爬虫，它们各自独立负责处理一个或多个指定的网址，在接收到爬虫引擎中的 response 响应后，负责对相应的网页内容进行解析，最终提取出相应的数据。

管道(Pipeline)负责接收从爬虫组件中提取出来的数据项 item，并对这些数据项进行相应的处理，常见的处理主要有：清洗数据、验证数据、检查重复数据、将数据导入数据库等。下载中间件(Downloader Middlewares)是存在于引擎和下载器之间的一个特定插件，它负责对引擎和下载器之间的通信进行处理。更重要的是，它还提供了一种方便的机制，能够以插入自定义代码的方式扩展 Scrapy 功能。

Scrapy 框架在实际工作中的数据流如图 2-4 所示:

(1) 调度器先根据队列里面的网址的优先级，将下一次要抓取的网址传递给引擎；

(2) 引擎收到调度器传过来的网址后，将网址传递给下载中间件；

(3) 下载中间件在收到引擎发过来的网址后，将对应网址传递给下载器；

(4) 下载器在收到网址后，通过互联网向对应网址发送 request 请求，进行网页的下载；

(5) 互联网中对应网址接收到下载器的 request 请求后，产生了对应的 response 响应返回给下载器；

(6) 一旦完成页面的下载后，下载器生成一个该页面的 Response 传送回下载中间件；

(7) 下载中间件在收到响应之后，与引擎进行通信，将该响应传递给引擎；

(8) 引擎接收 Response，并将 Response 通过爬虫中间件发送给爬虫。

(9) 爬虫处理 Response，将抓取到的数据项及新的 Request 请求通过爬虫中间件返回给引擎。

(10) 引擎将爬虫抓取到的数据项传递给管道，由管道负责对数据项进一步处理，同时将爬虫返回的 Request 请求返回给调度器，由调度器进行下一轮的调度。

(11)重复上述步骤直到调度器中没有更多的网址，则引擎结束工作。

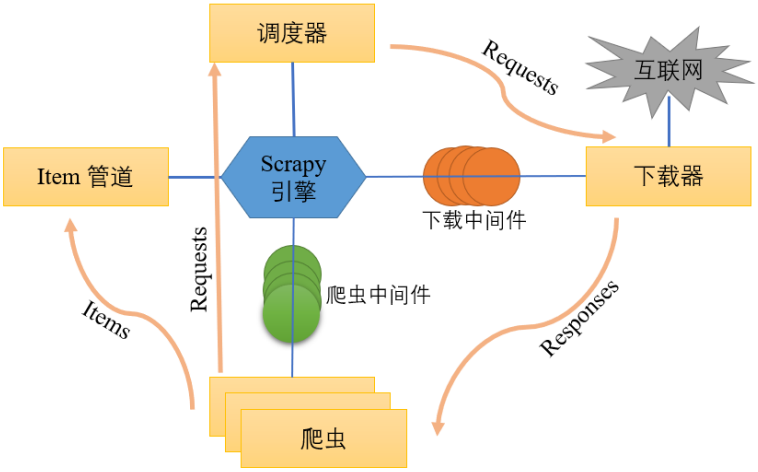

图 2-4 Scrapy 工作数据流

2.2 文本挖掘相关技术

2.2.1 信息量与信息熵

信息量主要用于对信息进行度量，它的影响因素为随机事件的概率；概率越小的事情发生时所产生的信息量就越大，因此一个具体事件的信息量应该是随着其发生概率而递减的，且不能为负。公式见(2-1)：

$$h(x) = -\log_2 p(x) \qquad \text{公式(2-1)}$$

信息量表达的是一个具体事件发生后所带来的信息，而信息熵[31]则是在结果出来之前对可能产生的信息量的期望。信息熵则是某一特定信息的概率，它反映的是信息在传输期间中的不确定性程度。对于接收者方来说，反映不确定性的度量值等价于平均信息量，因而将信息熵作为信息量的定量指标。公式(2-2)为离散型随机变量信息熵的计算表达式。

$$H(x) = -\sum_{i=1}^{n} p(x_i)\log(p(x_i)) \qquad \text{公式(2-2)}$$

从上面公式可以看出，信息熵依赖于概率分布$p(x)$，所以说信息熵$H(x)$的定义应概率的单调函数。信息熵具有单调性、非负性、累加性这三种性质。其中单调性指对于发生的可能性越高的事件，其拥有的信息量就越少；信息熵的非负性，即收到的一个信源号所包含的信息量应该是正值；而多个随机事件同时发生存在的总不确定性的度量值，可以表示成各个事件不确定性的度量值的累加和，这是累加性的一种体现。

2.2.2 TF-IDF 加权技术

TF-IDF[32]是一种基于统计方法的对特征进行加权的技术，它常用于信息检索和文本挖掘中，旨在评估一个单词对于语料集中的一个文档的重要程度。TF-IDF 值与单词在单个文档中出现的次数成正比，而与单词在整个语料库中出现的次数成反比，这将有助于过滤掉某些在语料库中高频出现并得到很高重视度的词。TF(Term Frequency)是词频，表示为公式(2-3)，$n_{i,j}$表示的是单词在文档 j 中出现的次数，$\sum_k n_{k,j}$是文档 j 中所有单词的个数；IDF(Inverse Document Frequency)是逆文本频率指数，表示为公式(2-4)，D 表示整个语料库的文档总个数，分母表示语料库中包含单词t_i的文档数。TF-IDF 值实际为 TF 与 IDF 的相互结果，见公式(2-5)，该算法的主要原则是：如果某个单词在同一文档中出现了较多次，但在其它文档中较少出现时，那么就认为这个词对该单一文档的重要程度较高，适合作为该文档的分类标准。

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}} \quad \text{公式(2-3)}$$

$$idf_i = \lg \frac{|D|}{|\{j: t_i \in d_j\}|} \quad \text{公式(2-4)}$$

$$tfidf_{i,j} = tf_{i,j} \times idf_i \quad \text{公式(2-5)}$$

根据公式(2-4)，如果某一个语料库中的文档数为 D，当中包含单词 t 的文档总数为 j，显然当 j 大的时候，得到的 IDF 值会变小，就说明单词 t 的区分能力不强；根据公式(2-5)，随着 IDF 值的降低，TF 值的表现将被削弱，从而使 TF-IDF 值降低。相反，当某个单词在同一文档中出现多次且在其它文档中较少出现时，那么它的 TF 值较高，且 IDF 值又较低，我们就认为这个单词可以较好地表达出该文档的特征，应该被赋予较高的权重，并将它们作为该文档的特征词用来区别于其它文档。

2.2.3 文本相似度算法

在数据挖掘和文本分析中，经常需要去评估文本数据之间差异的大小，实际需要衡量的是文本数据间的相似度[33]。常用的相似度算法有欧几里得距离、余弦相似度、皮尔逊相关系数等。

欧几里得距离，见公式(2-6)，指的是 n 维空间中两个点之间的实际距离，或该点到原点的距离（即向量的自然长度）。在平面几何或者立体几何中的距离，通常就是欧氏距离，因此欧氏距离也最容易理解。

$$dist_{ed}(x,y) = (\sum_{i=1}^{n}|x_i - y_i|^2)^{\frac{1}{2}} \quad \text{公式(2-6)}$$

余弦距离，见公式(2-7)，也称为余弦相似度，它以向量空间中两个向量夹角的余弦值作为标准来评估两个体数据之间差异的大小。夹角越趋近于 0 度，余弦值越接近于 1，即两向量的指向更加一致，则它们越相似；若两向量的方向完全相反，则夹角余弦值取最小值-1。当余弦值为 0 时，两向量正交，夹角为 90 度。因而可以看出，向量之间的余弦相似度与向量的幅值无关，只与向量的方向有关。

$$\cos(x,y) = \frac{<x,y>}{\|x\|\cdot\|y\|} \quad \text{公式(2-7)}$$

皮尔逊相关系数，用作为两个变量 X 与 Y 之间线性相关的统计度量，取值在-1 到 1 区间。若皮尔逊相关系数为 1，表示变量呈完全正相关，为 0 表示完全无关，-1 呈完全负相关。两个变量之间的皮尔逊相关系数定义如公式(2-8)，它表示变量 X 与 Y 之间协方差与标准差的商，两变量的标准差都不能为 0，否则皮尔逊相关系数将丢失定义。皮尔逊相关系数适用于：

(1) 两个变量都为连续数据，且相互呈线性关系。

(2) 两个变量在总体上呈正态分布，或者趋近正态的单峰分布。

(3) 两个变量的观测值成对出现且每对观测值之间互相独立。

$$p_{x,y} = cor(x,y) = \frac{cov(x,y)}{\delta x \delta y} = \frac{E[(x-x_\mu)(y-y_\mu)]}{\delta x \delta y} \quad \text{公式(2-8)}$$

2.2.4 Apriori 算法

在正式介绍 Apriori 算法之前，需要先对频繁项集挖掘算法[34]进行了解。频

繁项集挖掘算法是很多文本挖掘任务如关联规则、时间序列等的重要基石，它的具体实现过程是先定义一个判断候选集为频繁项的标准，通常使用最小支持度；接着计算满足最小支持度的频繁单项集或多项集，它是按照逐层进行的。

Apriori 算法[35]是频繁项集挖掘的一种具体实现，它支持查找等于或大于给定最小支持度的所有项集。Apriori 算法基于两个判断依据，一是若一个项集是频繁的，则它的非空子集也视作频繁项集。例如，若$\{A,B,C\}$被判断为频繁项集，则$\{A,B\}$、$\{A,C\}$、$\{B,C\}$、$\{A\}$、$\{B\}$、$\{C\}$也应该是频繁项集。二是如果一个项集不是一个频繁项集，那么它的超集都不是频繁项集，超集是与子集相对的概念。例如，若$\{A,B,C\}$是非频繁项集，那么$\{A,B,C,D\}$必然也是非频繁项集。

上面的两则判断依据也被视作为 Apriori 算法的先验性质，我们能够利用该算法的这些判断依据快速查找到频繁项。Apriori 算法的具体实现流程是：

(1) 输入数据集、最小支持度作为参数。

(2) 生成数据集中包含的所有单项候选集的集合C_1。

(3) 对C_1中的候选集进行扫描，若满足最小支持度则将候选集纳入集合L_1，否则将被去除。

(4) 对L_1集合进行组合以生成两项候选集的集合C_2。

(5) 重复(3)-(4)步骤，直至所有不满足最小支持度的项集都被去除。

2.3 特征工程与文本分类

本章节主要介绍的是第三章及第四章要用到的文本分类相关技术，具体包括文本预处理、文本特征提取、文本数据增强、文本表示技术和传统文本分类算法等内容。

传统机器学习分类算法用于处理文本分类任务需要经历的过程可以概括为：数据获取与分析、文本数据预处理、特征选择与提取、文本向量化表示和模型选择与训练，如图 2-5 所示。

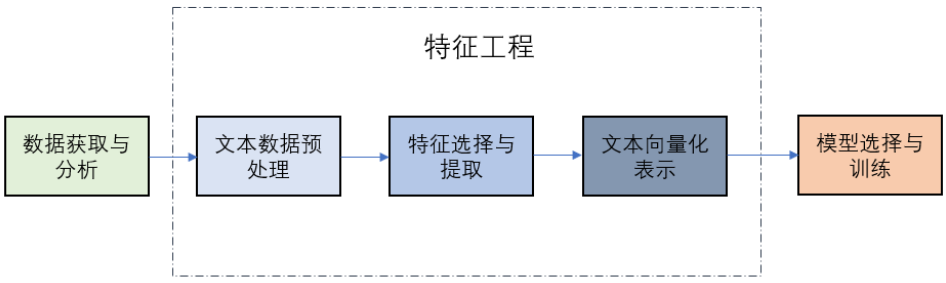

图 2-5 机器学习文本分类流程

从图 2-1 中明确看出，机器学习模型处理文本分类任务的一个完整流程为：首先获取数据并对数据进行分析，此步骤主要用于评估数据的量级，即样本与特征的规模及维度等；其次对文本数据进行预处理操作，从中进行特征选择与提取；再次将提取到的特征通过向量化进行文本表示；最后对经过特征工程处理后的文本数据选取合适的分类模型进行训练。下文将对文本分类任务用到的一些相关技术进行详细的赘述。

2.3.1 中文文本预处理

本小节主要介绍的是中文文本预处理技术[36]，本文研究的对象是中国在线食谱数据，绝大部分的数据都是中文，因此合理使用中文文本预处理技术至关重要。同时，中文文本的预处理相较于英文文本有很大不同。首先中文文本没有英文文本那样天然的空格分隔，无法直接使用空格或标点符号进行分词，因此通常需要使用分词算法来进行分词；其次是中文的编码通常是 Unicode 而非 UTF8，这就需要在分词的时候留意编码的问题。下面详细介绍几个常用的中文文本预处理操作。

首先是中文分词，它能够将中文语句按照语义切分成一个个词语，它是文本预处理中非常重要的基础部分，分词效果的好坏关系到了之后分类模型的准确率。分词算法主要有两种：基于词典的算法和基于统计的算法。基于词典的分词算法又称为机械分词算法，它的原理是根据一定的规则将未进行分词的字符串与经过初始化的足够大的词典进行匹配，若匹配成功，则该词就能够被分割出来。同时根据扫描方向的不同，基于词典的分词算法又分为正向匹配和逆向匹配；根据字符串不同的匹配长度可分为最大匹配和最小匹配。在实际应用中，通常将这上面几种方法进行组合使用，双向最大匹配法就是其中一个例子，它将正向最大匹配获取到的分词与逆向最大匹配法获取到的分词进行比较，进而选择正确的分词办法。基于统计的分词算法的原理是：每个句子中的词都是由字构成的，若相连的字在不同的文本里出现的次数越多的就越可能成为一个单词，所以我们说相邻字出现的频率能够评估它们组合成词语的概率，当这个出现的频率大于一定的阀值，就可以认为这些相邻字构成了一个单词。这类分词算法主要用到的统计模型有最大熵模型、N-gram、条件随机场 CRF、隐马尔科夫 HMM 等。综合对比这两种分词算法，基于词典的分词算法优点是速度快、实现简单，但对于存在歧义或是未登陆词的表现并不理想；基于统计的分词算法的优势在于它能够很好地解决上面分词算法出现的问题，但是缺点是处理分词的速度相对较慢。在实际应用中可以结合两种中文分词算法综合使用，以达到更好的分词效果。

其次是解决中文文本的编码问题，我们需要在进行中文文本预处理时进行规

范编码，可在存储数据时统一采用 UTF8；同时由于 Python2 不支持 Unicode 的处理，尤其是在代码里混合使用 Str 与 Unicode 时容易出现乱码问题，而 Python3 默认使用 UTF8 编码，因此更推荐使用这一版本。

再次是引入停用词，它能够有效减少储存空间的使用以及提高检索速度。停用词是在自然语言处理中需要被过滤掉的字或词的组合，它们是由人工进行整理而非自动生成的。对于中文文本的数据，我们可以使用一些权威机构或组织提供的已收集并整理好的中文停用词文档。

最后是数据清洗部分。由于每种数据的特点不同，针对数据进行数据清洗前需要分析脏数据的特征，进而灵活地选择相应的处理技巧。例如，对于通过爬虫采集的中文数据经常夹杂着一些 HTML 的标签，我们可以直接使用 Python 的正则表达式进行清洗。

2.3.2 文本特征提取

文本特征提取[37]是文本特征工程中的重要步骤，所提取特征的好坏直接影响了文本分类模型的效果。特征提取是针对特征项而言的，特征项是其所在目标文本的标识，能够将目标文本区分于其它文本。目前大部分中文文本分类模型都以单词作为特征项，并且这些单词特征项所构成的特征向量是高维的，它们导致了庞大的计算量。因此直接采用未经处理的特征向量进行文本分类是不现实的，这就需要完成特征提取这一个步骤，它的主要功能是在不影响文本语义信息的前提下尽可能地降低向量空间的维数，进而提升文本处理的效率。文本特征提取对于文本分类、聚类以及知识发现等领域具有不可忽视的作用。

在进行特征提取时，通过使用一些特征评估函数对向量空间中的特征项进行评分，然后根据评分值对特征项进行降序排列，抽取若干较高评分值的项作为特征，这就称为特征提取的过程。常用于构造特征评估函数的算法有：TF-IDF、互信息、期望交叉熵、二次信息熵、信息增益、主成分分析、N-gram 等。上述评估函数常见于各类文本挖掘任务中，它们各自有其自身的特点，每一种都有相应的选词标准，能从文本数据集中的所有单词中选择出有限范围的特征项。

2.3.3 文本数据增强

对于文本分类模型而言，我们希望能用于训练的数据量尽可能多且质量好，而数据增强[38]是一种能够有效扩展数据样本规模的方法。对于文本数据，数据增强常使用以下策略：

(1) 随机 shuffle 和 drop。shuffle 指对文本数据集进行顺序的打乱，drop 指对于文本的字或词随机选择删除；

(2) 同义词替换。随机地选择一些单词进行同义替换；

(3) 回译，也称为反向翻译。它是机器翻译中经常使用的策略，即对一段 A 语言文本翻译成 B 语言，然后再新翻译回 A 语言的过程。该策略已在 Kaggle 恶意评论分类竞赛中得以运用。

(4) 文档裁剪。通常一些较长文本的主要想法可能重复出现，通过对文本裁剪成多个子文本来得到更多数据。

(5) 生成对抗网络。GAN 通常用来生成图像数据，但有些研究使用了 GAN 生成文本数据。

2.3.4 文本表示技术

文本表示即运用向量化的技术将文本转变成计算机可以处理的数值形式，好的文本表示方法能给算法模型带来较大的提升。文本表示方法通常有离散表示和分布式表示两种类型。

文本离散表示的代表有 one-hot、TF-IDF、N-gram 等。其中 one-hot 是用于处理文本的传统表示方法，它将文本中的每个单词表示成一个长向量；向量的维度代表了文本的长度，该单词在向量中的某一个维度表示为 1，剩余表示为 0。one-hot 的使用十分简单，但其表示的向量通常高维高稀疏，且无视了词与词之间的语义和顺序。TF-IDF 是对 one-hot 表示的一种改进，它能够通过词频和逆文档频率对关键词进行抽取。

分布式表示通常将词转换成稠密且连续的定长向量，它克服了离散表示存在的一些缺陷，能够表示更多的信息。例如，它能够通过欧式距离或余弦距离等衡量出词与词之间“距离”的远近，进而来判断它们在语义上的联系。目前应用较为广泛的分布式词向量算法是 Word2vec[39][40]，它由 Google 公司在 2013 年进行了开源。Word2vec 采用两种不同的办法，分别是 Skip-gram 和 CBOW。其中，Skip-gram 是通过当前单词来预测上下文的概率，CBOW 则相反。

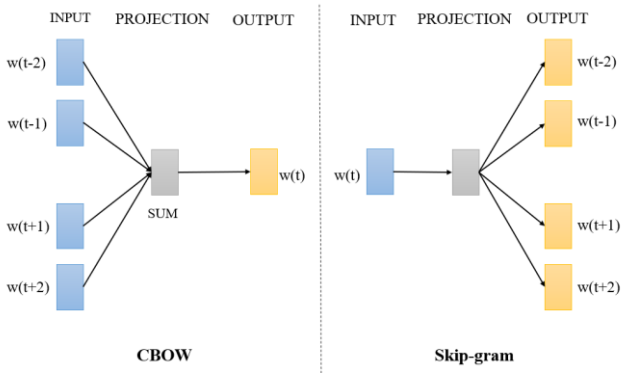

图 2-6 CBOW 和 Skip-gram 模型结构

2.3.5 传统文本分类算法

本小节主要介绍传统文本分类算法中的两种，分别是朴素贝叶斯和 K 近邻算法，同时还介绍了多标签分类任务的处理方式。

(1) 朴素贝叶斯

朴素贝叶斯是贝叶斯决策理论的组成部分，它是一种基于条件概率的分类方法。贝叶斯决策理论的核心内涵是选择出现概率最高的因素作为判断类别，即它是基于最高概率的决策理论。条件概率是概率论的内容，它计算的是基于一定条件下事件发生的概率，计算见公式(2-9)。另一种计算条件概率的策略是贝叶斯准则，它体现的是条件概率中条件与结果的交换，见公式(2-10)。

对于朴素贝叶斯分类算法[41]而言，给定一个输入项，它能够在该项出现的条件下分别去计算各个类别出现的概率估计值，进而得出分类结果。例如，在使用朴素贝叶斯算法对一个任务中的c_1、c_2两个类别进行判断时，我们可以预先定义好贝叶斯分类准则，见公式(2-11)。接着分别对 $p(c_1|x,y)$ 、$p(c_2|x,y)$进行计算，进而通过定义好的准则进行判断从而完成分类。

$$p(x|y) = \frac{p(xy)}{p(y)} \quad \text{公式(2-9)}$$

$$p(c_i|x,y) = \frac{p(x,y|c_i)p(c_i)}{p(x,y)} \quad \text{公式(2-10)}$$

$$class = \begin{cases} c_1,\ p(c_1|x,y) > p(c_2|x,y) \\ c_2,\ p(c_1|x,y) < p(c_2|x,y) \end{cases} \quad \text{公式(2-11)}$$

(2) K 近邻

K 近邻算法[42]是一种监督学习方法，同时它也是文本分类中常用的算法。它的工作原理是：给定一个训练样本，对于新输入的测试样本，能够基于距离度量方法在训练样本中找出与该测试样本最邻近的 K 个实例；若这 K 个实例的多数属于某一个类，那么就把该测试样本归属于这个类。具体的距离度量方法本文已经在 2.2.3 中介绍。

K 近邻算法没有显式的训练过程，它直接在测试过程中对测试样本和训练样本进行距离的计算，从而找到距离最近的 K 个样本，同时以多数投票的方式对这 K 个最邻近样本的类别进行判别。K 近邻算法的三要素分别是 K 值选择、距离度量、分类决策规则。其中，K 值较小时容易发生过拟合，过大则容易使近似误差增大；距离度量在文本分类中多采用余弦距离；分类决策规则常采用投票法或加权投票法。

(3) 多标签分类的处理方式

多标签分类任务通常有两种处理方式，一是尝试将多标签分类问题转变为单标签分类问题，二是直接通过改编算法进而实现多标签分类。

对于第一种方式，常见的转换策略有如下几种：

I. 二元关联法。做法是先将每个标签视作为一个单独的类，然后分别对这些单独类进行逐一训练。若一个训练集存在 5 种标签，那在二元关联中就会组合成 5 个不同的类进行拟合。这种方式没有考虑标签之间的关联性，局限性较大；

II. 分类器链法。在该策略下，假设X为特征空间，Y为标签空间，其中y_1，y_2，y_3为Y的子集。那么第一个分类器首先对(X，y_1）进行训练，第二个分类器对((X，y_1)，y_2)进行训练，以此类推组成分类器链。它考虑了标签之间的关联性，克服了第一种方式的缺陷；

III. 样本实例转换法。在该策略下，对每个样本所属的标签联合起来创建新的标签，从而使多标签问题转换成一个或多个单标签问题进行处理。

对于第二种直接改编算法的方式，它的性能较第一种要更加地优越。目前针对多标签分类而改编的算法有决策树、支持向量机、K 近邻等。

2.4 深度学习技术

2.4.1 深度学习概念

深度学习[43]常常与人工智能、机器学习等概念混淆，本质上来说，深度学习是对机器学习中的神经网络部分进行的深度拓展。深度学习包括了神经网络、深度神经网络以及深度增强学习。

过去十年对于深度学习的定义在不断发生变化，但大部分研究人员认为具有两层以上的神经网络才能称为深度学习。而神经网络的主要特点是：具有较多的神经元、负责的网络结构及连接方式、庞大的计算量以及能自动地完成高维特征的提取。目前深度学习技术的应用领域十分广泛，例如机器人、医疗诊断、图像处理、模式识别等。

2.4.2 卷积神经网络

卷积神经网络(CNN)[44]最主要的特点在于它能够利用它的多层次网络结构对数据的深层特征进行自动提取，并且不同层次的网络能够学到不同层次的特征。卷积神经网络主要由输入层、卷积层、池化层、全连接层和输出层构成。

卷积神经网络应用在图像领域中，可以识别二维图像的缩放、位移以及物体

形态扭曲等。与传统的人工神经网络相比，由于卷积神经网络引入了局部感知、权值共享和下采样三大核心思想，使得卷积神经网络在图像处理领域中大放异彩。其中局部感知是指每个神经元节点不在和下一层中的所有神经元节点相连，而是只与部分神经元节点相连，这样可以大大减少网络中的权值参数；权值共享是指对图像中有相同纹理特征的部分使用同一个权值参数，这样不但可以减少网络中的权值参数，而且还减少了冗余数据；下采样可以有效地减小数据图像矩阵的大小，这样可以有效地预防过拟合和更进一步地提取图像中的高维特征，其中常见的池化操作有均值池化和最大池化两种。这三大核心思想使卷积神经网络可以提取到图像的高维特征，并能够在有限的内存和规定的时间内完成庞大的计算。

卷积神经网络在自然语言处理中也得到了充分应用，本文以基于卷积神经网络处理文本分类问题为例，如图 2-7。输入层为 n*k 的句子矩阵，每一行是词向量，维度为 k，这里的输入可以类比为原始图像的像素点。接着进入不同尺度的卷积核进行卷积操作，用于提取特征。其次经过一个 Max Pooling 层，主要进行特征的压缩从而保留主要特征，它是通过 Pooling 操作能对每一个列向量的最大值进行提取并连接成为单行向量，从而实现特征的压缩和保留，同时该层还将每使非定长的句子转成定长的表示。最后一层为全连接层，它使用 softmax 分类器计算文本每个类别的概率并输出，进而得到分类的最终结果。

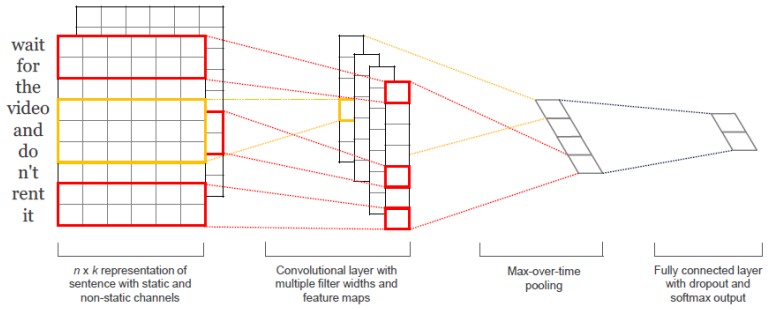

图 2-7 基于卷积神经网络的文本分类

2.4.3 循环神经网络

循环神经网络(RNN) [45]是一种通过递归连接形成记忆的网络。在前馈网络中，输入是相互独立的，但在 RNN 中所有的输入都是相互连接的，因此循环神经网络常用于处理动态的序列数据。同时，它还能随着时间推移不断地调整自身的网络结构，并对先前的序列数据保留一定的记忆能力。循环神经网络最主要的

目标是完成对序列数据的建模，序列数据常与时间的先后顺序相关联，而生活中大部分数据与时间都有关联，因此循环神经网络模型在深度学习中有着重要的地位。循环神经网络基本结构如图 2-8 所示。

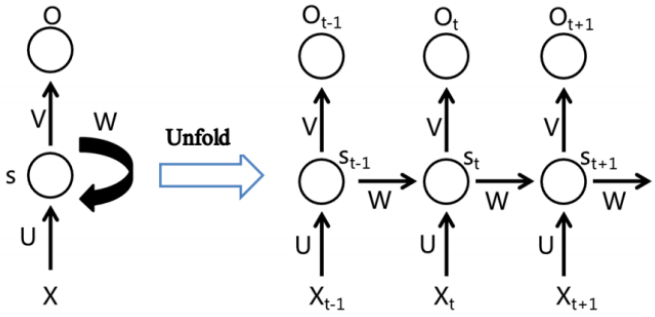

图 2-8 卷积神经网络基本结构

从上图可以看出，循环神经网络要求每一时刻都有相应的输入，即需要将一个序列在不同时刻上的数据依次输入到循环神经网络输入层中，而输出的结果可以是对序列当前时刻的预测，也可以是下一时刻的预测。

除了上述基本结构，还包括长短时记忆网络(LSTM)、双向循环神经网络(BRNN)以及深度循环神经网络(DRNN)等。其中双向循环神经网络的结构见图 2-9，它在时刻 t 处的输出不仅取决于序列中的先前元素，而且还取决于之后的元素。它是将两个独立的循环神经网络堆叠在一起，然后根据两个 RNN 的隐藏状态来计算输出结果。具体地，在任意时刻 t，输入会同时提供给两个方向完全相反的独立循环神经网络。它们独立进行计算，并且产生该时刻的最新状态和输出，最后将输出的结果进行拼接从而得到最终结果。

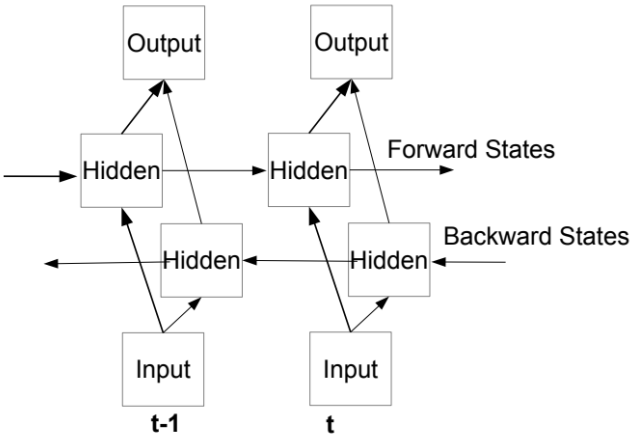

图 2-9 双向循环神经网络

2.5 本章小结

本章首先介绍了数据采集技术。其次介绍了文本挖掘相关技术，包括信息量与信息熵、TF-IDF 加权技术、文本相似度算法和频繁项集挖掘算法。再次，对特征工程与文本分类进行了详细介绍。最后介绍了深度学习的相关技术，包括深度学习、卷积神经网络和循环神经网络。

文本挖掘相关技术为下文的在线食谱数据的采集与分析提供了技术性的思路；同时，深度学习技术是自然语言处理的重要部分，通过对卷积神经网络和循环神经网络的介绍，为下文进行的基于食谱成分的多标签分类作铺垫。

第三章 在线食谱数据的采集与分析

在本章节，我们将运用 2.2 中所介绍的文本分析与挖掘相关技术对中国在线食谱展开多种维度的分析。我们首先在 3.1 中给出食谱数据采集模块的设计实现；3.2 主要分析了各菜系的成分多样性；3.3 主要工作是寻找各菜系的特色成分，找到人们常说的“地方特产”；在 3.4 中，通过研究所有食谱之间的相似性，从而期望找出其所属菜系之间的联系；3.5 的主要工作在于通过频繁项集挖掘算法，从所有食谱中找到经常使用的辅料集合。

3.1 在线食谱数据采集模块

3.1.1 爬虫模块的设计

为获取中国在线食谱网站的各大菜系的食谱数据，我们编写了 Scrapy 爬虫定向地从食谱网站中抓取食谱的标题、成分、口味、烹饪工艺、烹饪时间以及食疗标签等数据，并通过管道将数据存入 MongoDB 数据库，便于后期的使用。

目前在线食谱网站的反爬虫机制较为完善，对网站的访客有很多限制，例如用户代理限制，即网站的服务器能够识别用户所使用的浏览器版本及插件等。如果不对爬虫设置用户代理随机变换的话，则只能爬取到小部分的数据。

因此，我们需要先在 Scrapy 爬虫框架的 settings.py 中设置用户代理，具体设置详见算法 3-1。

Algorithm 3-1 设置用户代理

```
1. BOT_NAME = 'snack'
2. SPIDER_MODULES = ['snack.spiders']
3. NEWSPIDER_MODULE = 'snack.spiders'
4. # Crawl responsibly by identifying yourself (and your website) on the user-agent
5. USER_AGENTS = [
6. "Opera/9.80 (Macintosh; Intel Mac OS X 10.6.8; U; fr) Presto/2.9.168 Version/11.52",
7. "Mozilla/5.0 (Windows; U; Windows NT 5.1; zh-CN) AppleWebKit/523.15 (KHTML, like Gecko, Safari/419.3) Arora/0.3 (Change: 287 c9dfb30)",
```

同时我们需要在中间件(middleware)中开启随机变换用户代理的功能，这样我们才能在爬取数据的过程中进行用户代理的自动变换，以达到伪装成不同浏览器的作用，具体实现见算法 3-2。

Algorithm 3-2 设置中间件

```
1.  class RandomUserAgent(object):
2.      # 根据预定义的列表随机更换用户代理 User-Agent
3.      def __init__(self, agents):
4.          self.agents = agents
5.      @classmethod
6.      def from_crawler(cls, crawler):
7.          return cls(crawler.settings.getlist('USER_AGENTS'))
8.      def process_request(self, request, spider):
9.          request.headers.setdefault('User-Agent', random.choice(self.agents))
```

为了将抓取的数据存入 MongoDB 数据库，我们还需要对 Scrapy 爬虫框架中的管道(pipelines)进行设置，具体实现如 3-3。

Algorithm 3-3 设置管道

```
1. def process_item(self, item, spider):
2.     valid = True
3.     for data in item:
4.         if not data:
5.             valid = False
6.             raise DropItem('Missing{0}!'.format(data))
7.     if valid:
8.         self.collection.insert(dict(item))
9.     return item
10. def __init__(self):
11.    connection =MongoClient(
12.        settings['MONGODB_SERVER'],
13.        settings['MONGODB_PORT'])
14.    db = connection[settings['MONGODB_DB']]
15.    self.collection = db[settings['MONGODB_COLLECTION']]
```

综上，我们设计了在线食谱数据采集模块，该爬虫模块的结构图如 3-1 所示。

第 1 步采用不断变换用户代理的方式访问在线食谱网站；

第 2 步维护一个待抓取的 URL 网页列表，默认会从人工设置的第一个起始网页开始；

第 3 步从待抓取的 URL 列表中取出一个网址，根据相应的网页字段进行解析，并通过正则表达式进行数据的清洗和过滤；

第 4 步将解析出来的数据与 Scrapy 中定义的 item 进行封装；

第 5 步将封装好的食谱数据导入 MongoDB 数据库中；

第 6 步判断下一次待抓取的 URL 是否为空，为空则退出爬虫程序；否则继续数据爬取。

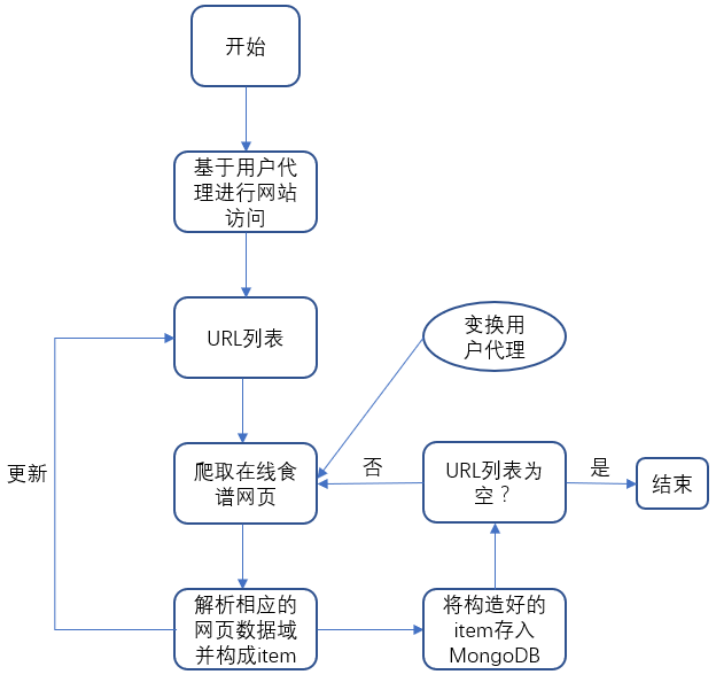

图 3-1 Scrapy 爬虫模块设计

3.1.2 数据清洗与统计

数据清洗是数据预处理的重要部分，它在本文中的主要作用是删除食谱成分数据中的干扰项以及重复数据。

对于本文所采集到食谱数据，它本身携带了一些标点、数字、HTML 标签等杂项，例如在食谱成分数据中出现频率较多的是''标签，这些都属于数据中的噪声，因此需要进行过滤。我们采用 Python 的 re 库，该库提供了丰富的正则表达式功能，具体实现见算法 3-4。

Algorithm 3-4 数据清洗

```
1. import re
2. def parse_item(self, ingredient):
3.         # 去掉'<span>成分</span>'的标签
4.         pattern_a = '<span>'
5.         pattern_b = '</span>'
6.         temp = re.sub(pattern_a, '', ingredient)
7.         result = re.sub(pattern_b, '', temp)
8.         result = re.sub(r' [0-9] ', ' ', result)    #删除数字
9.         result = re.sub(r'[{}]+'.format('!,;:?"\''),'',text)    #删除标点
10.        return result
```

在完成了对食谱数据的采集以及清洗工作之后，本章的后面章节将详细对这些食谱数据进行多种维度的数据分析。因此，这里我们仅对预处理后的数据集进行初步统计，详见表 3-1。

表 3-1 数据集统计

样本集	食谱数/条	合计/条
中华菜系	13658	35782
家常菜系	18056	
中国小吃	4068	

3.2 在线食谱的成分多样性分析

3.2.1 问题描述与实验步骤

探索各菜系的成分多样性的问题，旨在解决如下两个问题：

(1) 每个菜系所在地区的人们在完成一道食谱时，平均用到了多少种不同的成分？我们将这个问题理解为成分消费的多样性；

(2) 一个地区的菜系在成分组合上会有多大的差异？换句话说，同一菜系的不同食谱所包含的成分会有多大的不同，即成分组合的多样性。

通过上面的介绍，我们需要厘清成分消费多样性与成分组合多样性的不同

点。首先，成分的消费多样性指的是某个菜系中平均每道食谱用到的不同成分的种类数，如果某个菜系平均一道食谱用到的成分种类数较高，我们认为该菜系的成分消费多样性较高，体现的是局部特点；而成分的组合多样性是针对于整个菜系而言的，如果同一个菜系中的不同食谱之间所用到的成分类型差异较大，我们认为这个菜系的成分组合性是相对较高的，即该菜系食谱整体用到的成分种类较多。一个菜系的成分多样性，能够反映在成分消费的多样性以及成分组合的多样性。它们可能受到多种因素的制约，例如地理位置，农业情况，气候环境及习俗文化等。

上文对成分多样性的问题进行了分析，将菜系的成分多样性问题细分成了成分的消费多样性和组合多样性。下面本文就这两方面分别进行实验，考虑到大多数中国食谱的辅料成分仅用于调味作用，这里我们主要研究的是主料成分的多样性，因此该实验用到的成分全部为食谱的主料成分。首先对成分消费多样性进行计算，分为两个步骤：

(1) 由于每个中国菜系的食谱数都是不一样的，我们固定好每个菜系随机取60 道食谱，并且要求每道食谱至少包含一个成分；这样我们就得到了 13 个不同菜系的成分集，每个菜系包含相同的食谱数；
(2) 统计各中国菜系平均每道食谱所消费的成分种类数；
(3) 对实验的结果进行整合，并可视化。

其次基于信息熵公式对成分组合的多样性进行计算，因为信息熵反应了系统的不确定性，当一个菜系的成分分布信息熵较高时，就意味着该菜系食谱有着更加丰富的成分组合。具体实验步骤如下：

(1) 将每个菜系视作其包含的所有成分组成的一个概率分布，计算每个菜系中的每类成分出现的总次数，并将它们进行归一化，这样就得到每个中国菜系的成分分布。
(2) 通过信息熵公式(2-2)对成分分布进行计算，该公式中，变量p定义为每种成分x_i在所在菜系分布的频率。

3.2.2 实验结果分析

食谱成分的消费多样性和组合多样性的实验结果如图 3-2、3-3 所示，我们可以清楚地看到，东北菜的成分消费多样性最高，而广东菜的成分组合多样性最高。这意味着，与广东菜相比，东北菜平均每道菜用到的成分种类数更多，但其整个菜系所拥有的成分种类数较少。同时我们还可以从图中看到，成分的消费多样性和组合多样性有很高的相似性。这意味着成分消费多样化性较高的菜系往往具有较高的成分组合多样性，而成分消费多样性较低的菜肴也有较低的成分组合

多样性。之所以出现这种情况，可能的原因是随着一个菜系的成分组合多样性的增大，人们可以选择更多不同的成分来完成一道食谱，从而可以制作出相对独特的食谱，那么单个食谱所包含的成分种类也相对较多，即成分消费多样性较高。

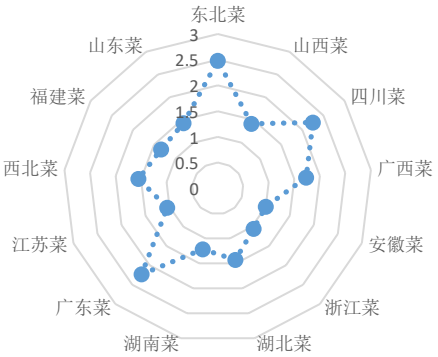

图 3-2 成分消费多样性

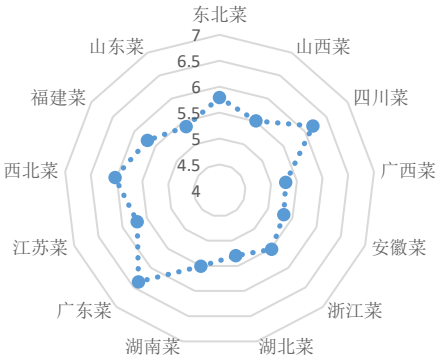

图 3-3 成分组合多样性

3.3 在线食谱的特色成分分析

3.3.1 问题描述与实验步骤

由于食谱的成分可能产自不同的地方，有些成分是各地通用的，例如土豆、西红柿等；有的则是具有地方特色的。我们称后者为具有特色的成分，它能作为一种标识去代表的相应菜系。换言之，特色的成分应该只在特定的菜系中被集中使用，而在其他菜系中较少被使用。通过研究在线食谱的特色成分，帮助我们去了解每个菜系的地方特性。具体而言，特色的成分应该具备如下要求：

(1) 集中地出现在特定菜系的多数食谱中；

(2) 较少地出现在其它菜系的食谱中。

TF-IDF(词频-逆文本频率指数)是常用在数据挖掘领域中的一种加权技术，它可以反映某一个字词在一个文件语料库中对于某份文档的重要程度，如果一个字词在某个文档中的频率较高，而在其他文档出现的频率较低，那么就认为该字词具有较高的区分度，能够用于分类。

结合上文关于特色成分的具体要求，我们选择 TF-IDF 用以计算每个菜系中的成分所对应的权值，具体的实验步骤如下：

(1) 在数据预处理中，各菜系分别构成一个个独立的文档，每个文档包含了所在菜系的所有成分，文档中的每个成分视作一个最小单元；我们最终获得了 20 份不同菜系组成的文档集合。

(2) 使用 TF-IDF 对这些菜系文档进行计算，得到每个菜系文档中相应成分的权值，然后降序进行排列。

(3) 通过这些成分的权值，我们运用词云图对西北菜系和湖北菜系进行可视化，如图 3-4、3-5 所示。

图 3-4 西北菜特色成分

图 3-5 湖北菜特色成分

3.3.2 实验结果分析

图 3-4、3-5 分别显示了西北菜和湖北菜中权值最高的 100 个显著成分。词云图中的成分字体越大，相应的权值也越大，即在所有中国菜系中的成分文档集合中更有区分度。比如西北菜的“羊肉”(mutton)和湖北菜的“武昌鱼”(blunt-snout bream)，在它们在各自菜系中的权值最大，可以视作为特色的成分。事实上，湖北省的全省面积中，仅是平原湖区就占了百分之二十的面积，它属长江水系，水资源较为丰富；而西北区有广阔的草原，囊括了新疆草原、青海草原两大草原。这可以解释为什么属水产品的武昌鱼在湖北菜系中更加流行，而属牲畜肉类的羊肉在西北菜系中占主要地位。

3.4 在线食谱的复杂性分析

3.4.1 问题描述

影响食谱复杂程度的因素有很多，例如成分的数量、烹调方法和烹饪时间等。因此，对于中国在线食谱的复杂性分析，可从研究各菜系食谱的主料及辅料成分的数量分布、烹饪时间分布及平均烹饪时长、烹饪工艺分布这几方面着手。在这一节中，我们将从这三个方面分析各菜系在线食谱的复杂性。

3.4.2 实验及结果分析

在成分数量方面上，一道食谱用到的成分越多，我们就认为它越复杂。由于食谱中的成分由主料成分和辅料成分组成，因此，我们对各个菜系食谱分别从主料成分和辅料成分两个方面进行了分析。经过统计，我们分别得到了中国各大菜系中主料成分和辅料成分的分布情况。图 3-6 显示了八大中国菜系各自的主料成分分布。从主料成分数量分布中，我们可以看到，数量只有 1 种主料成分的食谱所占的比例最高；从包含有 2 种以上主料成分的食谱比例来看，广东菜比其他菜系更加复杂。图 3-7 显示了中国各大菜系辅料的数量分布情况。从各个菜系中占比例最高的一项来看，我们可以清楚地看到四川菜中的辅料成分数量最多，高达 9 种；而山东菜中的辅料成分数量最少，仅为 2 种。

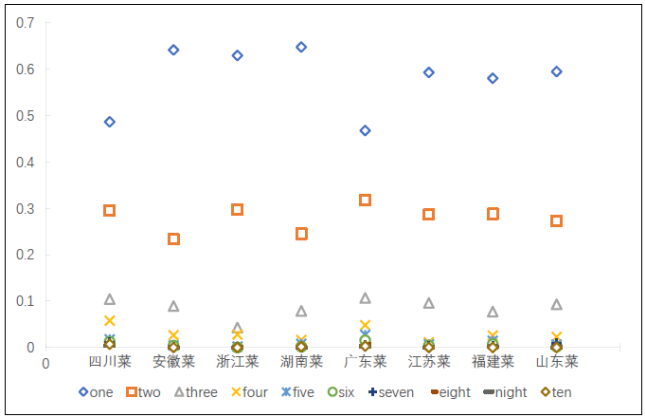

图 3-6 主料成分分布

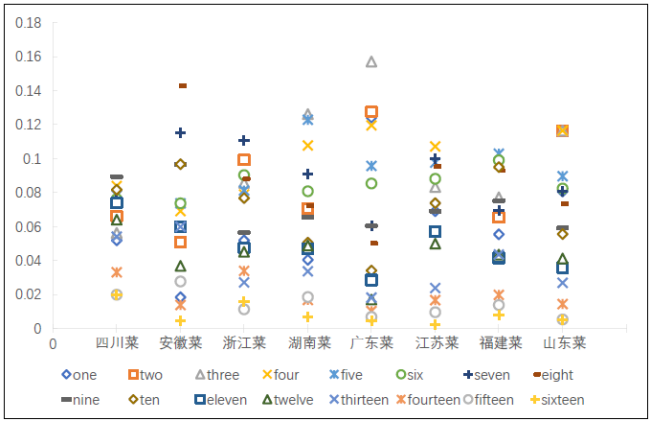

图 3-7 辅料成分分布

在烹饪时间方面上，一道食谱的制作耗时长短，能够反映食谱的复杂性；制作的耗时越长，相应就越复杂。因此，我们从烹饪时间的角度来分析各大菜系的复杂性。通过统计，我们得到了八大中国菜系食谱在制作时间上的分布，结果如

图 3-8 所示。同时我们计算出了各大菜系食谱的平均制作时间，结果如图 3-9 所示。从图 3-8 分布中占比最高的时间项来看，四川菜、广东菜、江苏菜、福建菜的烹饪时间最长，接近 30 分钟；而浙江菜和山东菜的烹饪时间最短，不到 10 分钟。需要特别注意的是，烹饪时间为 30 分钟到 120 分钟区间的食谱在江苏菜系中占很大比例。这可以解释为什么在图 3-9 江苏菜的平均烹饪时间是最长的，接近 40 分钟。

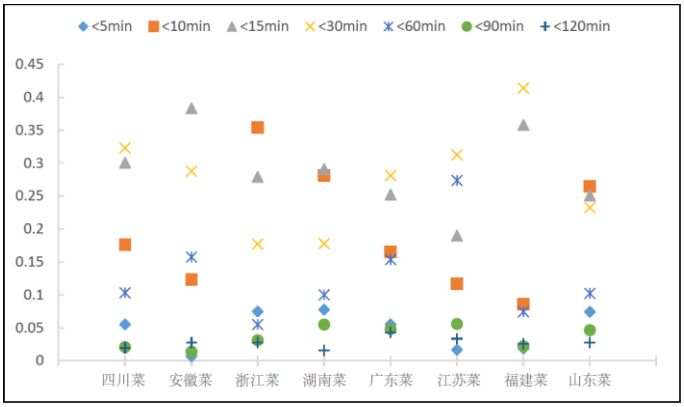

图 3-8 各菜系的食谱制作时间分布

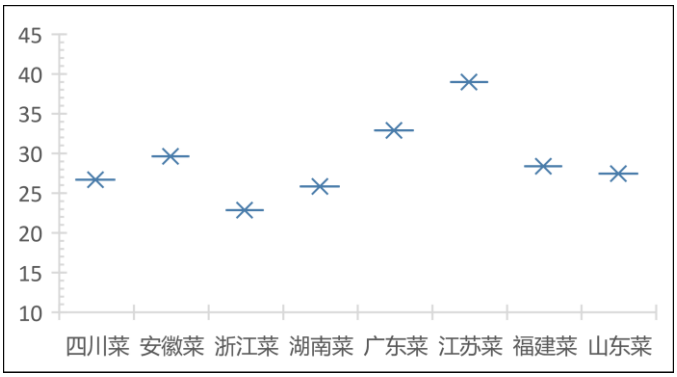

图 3-9 各菜系的单位食谱平均制作时间

在烹饪工艺方面上，中国历史上已经产生了多达 36 种基本的烹饪方式，它们完成的难易程度反映着不同程度的食谱复杂性。经过烹饪过程后，多种成分转换成了饭桌上的菜肴，同时也使营养成分容易被人体吸收。通过统计，我们得到了各个菜系食谱的烹饪工艺的数量分布，如图 3-10 显示。我们可以清楚看到，四川菜在“炒”工艺中所占的比例高于其他菜系，其次是湖南菜和山东菜。通过“炒”的工艺可以快速制作出一道菜，这就可以说明图 3-9 中四川菜、湖南菜和山东菜平均烹饪时间较短的原因。同时，“煲”的工艺在广东菜中所占的比例最高，该工艺需要消耗更长的时间，这就解释了为什么广东菜在平均烹饪时间上相对较长。

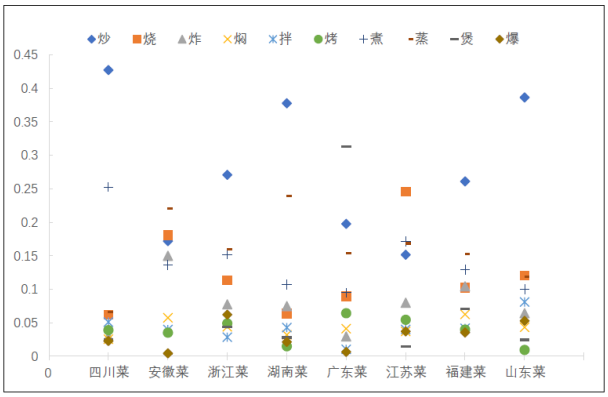

图 3-10 各菜系烹饪工艺的分布

3.5 在线食谱的相似性分析

3.5.1 问题描述与实验步骤

本文基于食谱的成分、口味和烹饪工艺这三个方面，来衡量不同食谱之间的相似性。我们以四川、湖南、广东、福建、浙江、江苏、安徽、山东等地组成的八大菜系作为研究对象，通过对食谱之间的相似性进行分析，进而探索八大菜系之间的关联性，是本节的研究意义所在。

在对食谱相似性进行计算时，我们考虑使用余弦相似度算法，它通常在数据挖掘任务中用于度量两个向量之间的相似性，两个食谱向量之间夹角的余弦值能帮助确定这两个食谱向量是否指向相同的方向，进而衡量它们的相似度。在本实验中，具体步骤如下：

(1) 将每个食谱表示为一个向量，向量的元素分别为食谱的成分、口味及烹饪工艺，并对向量进行归一化，使向量的元素之和为 1，如此便得到了一组能够表示食谱特征的向量；

(2) 将所有的食谱向量转换到向量空间，并应用余弦相似度计算任意两个食谱向量之间的相似性；

(3) 将计算得到的食谱相似度结果保存，并应用 d3.js 力导向图[46]进行可视化，如图 3-11 所示。

3.5.2 实验结果分析

在图 3-11 中，我们可以清楚看到，每个节点代表一个食谱，不同颜色的节点团簇代表着不同的菜系。两个食谱节点之间连线的长度与它们之间相似度值的大小成反比，即两个节点之间的连线长度越短，这两个食谱之间的相似度就越高，

反之亦然。

从图中还可以观察出属于相同菜系的食谱之间相似度较高，因此连接更加紧密。川菜和湘菜的食谱连线较为紧密，形成了一个簇。事实上川湘地区在位置上邻近，我们推测地理位置上的相近对食谱相似性可能有着重要的影响。此外，我们注意到，江苏菜系和浙江菜系的食谱节点连接也比较紧密。而江苏跟浙江地区在文化上有着密切联系，有着同样的江南文化，进而推测文化的相似性也可能会对不同菜系的食谱相似度产生重要的影响。

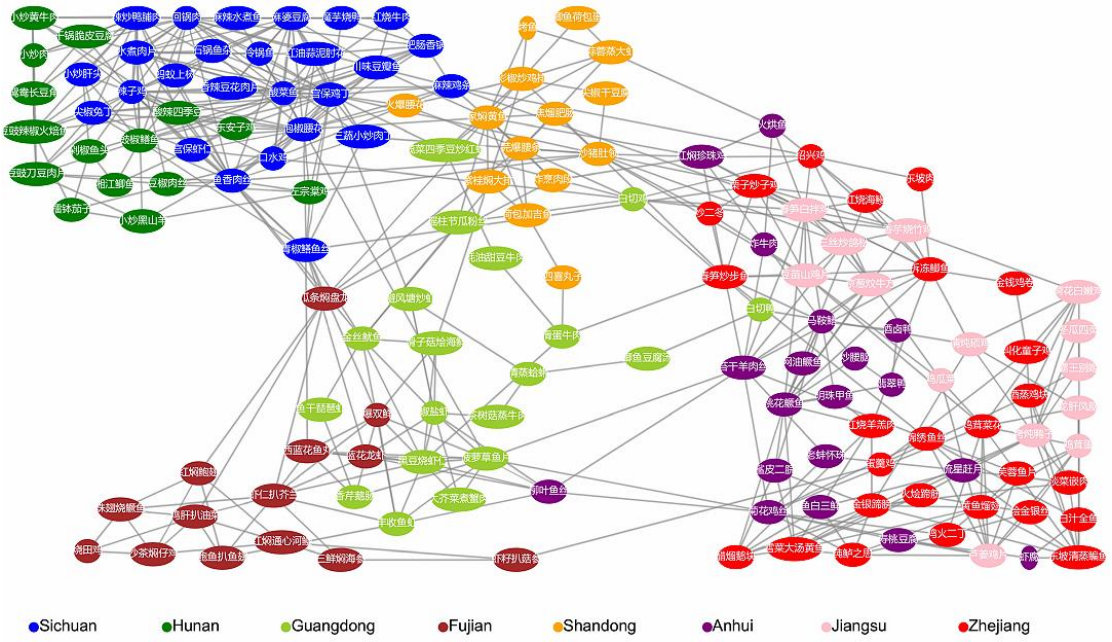

图 3-11 各菜系食谱的相似性

3.6 在线食谱的惯用辅料分析

3.6.1 问题描述与实验步骤

中国菜系植根于中国传统文化，烹饪的过程体现了自然和平衡的原则，体现在进行食谱制作时，人们总是高度重视配料成分的合理搭配。每道菜通常都由一到两种辅料成分组成。事实上，食谱的口味主要取决于盐、糖和酱油等辅料成分的组合搭配。因此，探索辅料成分的惯用组合有助于我们了解中国饮食文化。

为了研究辅料成分的惯用搭配，我们采用 Apriori 算法从食谱中找出辅料成分的频繁项。我们对中国各个菜系所有食谱的辅料成分数据集上应用 Apriori 算法。Apriori 算法将根据用户支持的最小值来寻找频率项集。具体过程如下：

(1) 我们将最小支持度设置为 0.1 和数据集$D = \{T_1, T_2, T_3, T_4, \ldots, T_n\}$。

(2) Apriori 算法首先构造候选 1 项集.然后，如果候选 1-项集的支持值低于最小支持值，则通过剪枝生成频繁的 1-项集。

(3) 在此基础上，算法构造候选 2-项集，并从候选 2-项集里剪除一些不频繁的项集，生成频繁的 2-项集。

(4) 上述过程将被重复，直到不再创建候选项集为止。

3.6.2 实验结果分析

图 3-12 显示了我们的实验结果。我们可以清楚地看到 Apriori 算法产生了 31 个频繁的 2-项辅料集、19 个频繁的 3-项辅料集和两个频繁的 4-项辅料集。所有频繁项都为辅料成分的惯用搭配。结果表明，生姜、葱、生抽和盐是中国菜系中最受欢迎的辅料成分。上述四种辅料成分通常组合在一起，它们的组合项占 11.83%。另外有 24.16%的食谱使用葱、生姜和食盐作为辅料组合，同时有超过三分之一的食谱选择生姜和食盐作为辅料组合。通过对各大菜系的辅料成分进行频繁项的挖掘，我们探索了中国食谱在辅料成分组合上的特点。

L2	support	L3	support	L4	support
[生姜, 食盐]	32.42%	[葱, 生姜, 食盐]	24.16%	[葱, 生姜, 食盐, 生抽]	11.83%
[葱, 食盐]	31.38%	[葱, 生姜, 生抽]	15.47%	[葱, 料酒, 生姜, 食盐]	10.70%
[葱, 生姜]	30.32%	[生姜, 食盐, 生抽]	14.78%		
[食盐, 生抽]	25.75%	[葱, 食盐, 生抽]	14.71%		
[料酒, 食盐]	20.18%	[料酒, 生姜, 食盐]	13.64%		
[生姜, 生抽]	19.99%	[葱, 料酒, 生姜]	12.75%		
[葱, 生抽]	19.51%	[葱, 料酒, 食盐]	12.34%		
[白砂糖, 食盐]	18.60%	[葱, 白砂糖, 生姜]	12.07%		
[味精, 食盐]	18.44%	[葱, 白砂糖, 食盐]	11.23%		
[大蒜, 食盐]	17.36%	[白砂糖, 生姜, 食盐]	11.22%		
[植物油, 食盐]	16.77%	[白砂糖, 食盐, 生抽]	11.19%		
[料酒, 生姜]	16.52%	[大蒜, 生姜, 食盐]	11.17%		
[白砂糖, 生抽]	15.92%	[葱, 大蒜, 生姜]	10.71%		
[白砂糖, 生姜]	15.13%	[葱, 食盐, 味精]	10.63%		
[大蒜, 生姜]	14.90%	[味精, 生姜, 食盐]	10.56%		
[淀粉, 食盐]	14.87%	[白砂糖, 生姜, 生抽]	10.44%		
[葱, 料酒]	14.82%	[葱, 白砂糖, 生抽]	10.28%		
[葱, 白砂糖]	14.82%	[葱, 大蒜, 食盐]	10.23%		
[葱, 大蒜]	13.61%	[葱, 植物油, 食盐]	10.06%		
[胡椒, 食盐]	13.00%				
[料酒, 生抽]	12.73%				
[大蒜, 生抽]	12.45%				
[芝麻油, 食盐]	12.34%				
[葱, 味精]	12.17%				
[葱, 植物油]	12.14%				
[植物油, 生姜]	11.97%				
[生姜, 味精]	11.92%				
[鸡精, 食盐]	11.17%				
[调和油, 食盐]	10.15%				
[植物油, 生抽]	10.05%				

图 3-12 各菜系食谱的惯用辅料

3.7 本章小结

本章主要是基于中国各大菜系的食谱数据，对中国在线食谱展开了详细的数据分析，充分地挖掘了食谱的成分、口味、烹饪时间、烹饪工艺方面的信息。通过一系列的实验，我们发现相邻菜系地区的食谱有着较强的相似性，以及成分消费多样性和成分组合多样性的相关性较强。我们还发现，某些成分独特地代表了一种特定的菜系。此外根据各大菜系食谱的烹饪时间，烹饪方式和成分的数量，我们还分析了它们的复杂性。最后，我们还探索了在中国菜系中经常使用的一些辅料成分的组合，它充分反映了一些特定辅料成分对于中国菜系的重要程度。

第四章 基于食谱成分的多标签文本分类

在本章，我们的研究对象为食谱中携带的标签。首先，我们对中国八大菜系的标签分布 TOP-5 进行了统计，详见图 4-1。从图中的结果可以看出，各个菜系的食谱所携带标签存在一些相似性，例如标签“防癌”在各大菜系食谱中出现的占比排名均在前两名。同时，通过结果我们可以看到各地区用户对于食谱标签的关注偏好，因此对食谱标签的合理分类将有助于用户找到更合适的食谱。

菜系	标签	占比	菜系	标签	占比
四川菜	防癌	6.87%	福建菜	防癌	6.63%
	助消化	4.30%		抗氧化	4.47%
	健体	3.94%		健体	3.98%
	免疫力	3.48%		降血压	3.11%
	抗氧化	3.16%		补血气	3.06%
湖南菜	助消化	7.45%	安徽菜	防癌	5.72%
	防癌	7.24%		抗氧化	4.17%
	健体	4.32%		健体	3.71%
	抗氧化	4.02%		补血气	3.19%
	补血气	3.01%		健脑	3.04%
山东菜	防癌	7.22%	广东菜	防癌	6.09%
	抗氧化	4.35%		健体	3.64%
	健脑	3.42%		抗氧化	3.64%
	助消化	3.36%		降血糖	2.64%
	健体	3.24%		降血压	2.55%
江苏菜	防癌	7.25%	浙江菜	防癌	8.14%
	健体	3.81%		抗氧化	4.98%
	抗氧化	3.59%		健体	4.94%
	健脑	3.48%		健脑	3.94%
	助消化	3.23%		降血压	3.57%

图 4-1 中国在线食谱中八大菜系的标签分布 TOP-5

多标签文本分类是自然语言处理中的重要内容。本章的目标是基于在线食谱的成分构建出性能良好的多标签分类模型，该模型的建立将有助于缓解人工维护标签体系的压力，帮助完整标签自动分类。为了保证输入到模型中的食谱样本的质量，我们对所有数据集进行过滤，只保留完整包含成分和标签数据的食谱样本。最终，本章实验所用到的食谱样本共计 15,747 条，食谱标签共计 447 种。

本章节在 4.1 中给出了传统机器学习在食谱多标签分类任务中实现，具体运用了贝叶斯和 K 近邻算法；在 4.2 中，我们以 word2vec 训练的词向量作为输入，并使用了深度学习模型实现了基于食谱成分的多标签分类器，具体应用了卷积神经网络(CNN)和循环神经网络(RNN)；在 4.3 我们介绍了本次实验的环境以及对 CNN 和 RNN 模型进行了调参，并运用了数据增强策略，紧接着对各个多标签分

类模型进行了性能对比和分析；最后，对本章内容进行总结。

4.1 传统机器学习算法的实现

在使用深度学习模型完成食谱多标签分类任务之前，我们需要进行基准实验，在本节我们将在 Scikit-learn 平台上实现贝叶斯、K 近邻算法的多标签分类。

4.1.1 输入数据预处理

本小节主要对输入模型的食谱数据进行预处理，包括食谱成分数据的特征工程和食谱标签数据的向量化。

(1) 食谱成分数据的特征工程

本文的基准实验将采用 Scikit-Learn 平台实现机器学习的文本分类模型，我们首先需要对输入的食谱成分数据进行特征工程。在 2.3.2 已经对特征提取技术进行了介绍，同时在 3.1.2 处对食谱数据进行了清洗，这里我们拟采用 TF-IDF 技术对输入数据进行特征提取。为了统一向量空间，我们将训练及测试的食谱成分数据联合输入到 Scikit-Learn 提供的 TfidfVectorizer 库函数进行特征工程向量化，这里的输入数据包含了食谱的主料成分及辅料成分。在全部样本进行向量化之后，我们对训练样本及测试样本进行区分。具体实现见算法 4-1：

Algorithm 4-1 食谱成分特征工程

```
1. def tfidf_transform(db, train_count):
2.     from sklearn.feature_extraction.text import TfidfVectorizer
3.     vector = TfidfVectorizer()
4.     # 对输入样本统一转成 TF-IDF 向量空间
5.     tf_idf = vector.fit_transform(db)
6.     # 切分输入集和测试集
7.     return tf_idf[:train_count], tf_idf[train_count:]
```

(2) 食谱标签数据的向量化

对于输入到机器学习模型的标签数据，需要对它们进行向量化。本文的食谱数据总共包含 447 种标签，因此经过向量化后每个食谱对应的标签维度为 447 维。鉴于本文需要进行的是多标签分类任务，我们采用 Scikit-Learn 提供的 MultiLabelBinarizer 库函数对食谱标签数据进行向量化，接着对训练标签和测试标签进行区分。具体实现见算法 4-2：

Algorithm 4-2 食谱标签向量化

```
1. def flag_process(flag_db, train_flag_count):
2.     from sklearn.preprocessing import MultiLabelBinarizer
3.     # 将训练及测试样本的标签统一进行向量化
4.     Y = MultiLabelBinarizer().fit_transform(flag_db)
5.     # 划分训练和测试样本的标签集
6.     return Y[:train_flag_count], Y[train_flag_count:]
```

4.1.2 贝叶斯分类模型

本文要进行的多标签分类，而贝叶斯分类是一个二分类模型。结合 2.3.5 中对于多标签分类的处理方式，我们拟采用二元关联法，即组合多个贝叶斯二分类模型以实现多标签分类。在本文中，所有食谱数据共包含 447 类标签，因此我们需要关联 447 个二元分类器，最后综合所有分类器的结果得到最终的分类结果。本文将采用 sklearn 平台提供的 OneVsRestClassifier 函数对贝叶斯分类器进行二元关联，具体的实现见算法 4-3：

Algorithm 4-3 贝叶斯分类器

```
1. def bayes_classifer(x_train, x_test, y_train, y_test):
2.     from sklearn.multiclass import OneVsRestClassifier
3.     from sklearn.naive_bayes import GaussianNB
4.     classifier = OneVsRestClassifier (GaussianNB())
5.     # 对输入训练样本进行训练
6.     classifier.fit(x_train, y_train)
7.     # 对输入测试样本进行预测
8.     predictions = classifier.predict(x_test)
9.     return predictions
```

4.1.3 K 近邻分类模型

K 近邻算法已在前面进行了介绍，它是经典的机器学习分类算法。同样它本身是一个二元分类算法，结合 2.3.5 多标签分类的处理方式，我们拟采用 K 近邻的改编算法以实现基于食谱成分的多标签分类任务。

我们采用开源的 skmultilearn 库提供的 MLkNN 函数，它在底层通过改编

KNN 算法以满足多标签分类需求。具体的实现见算法 4-4：

Algorithm 4-4　K 近邻分类器

```
1.  def knn_classifer(x_train, x_test, y_train, y_test):
2.      from skmultilearn.adapt import MLkNN
3.      classifier = MLkNN(k=5)
4.      # 对训练样本进行训练
5.      classifier.fit(x_train, y_train)
6.      # 对测试样本进行预测
7.      predictions = classifier.predict(x_test)
8.      return predictions
```

4.2 深度学习分类模型的实现

本节要进行的是深度学习分类模型的实现，主要内容有训练词向量、输入数据预处理、基于 CNN 的分类模型，基于 RNN 的分类模型。

4.2.1 训练词向量

词向量，也称词嵌入(word embedding)，它在自然语言处理中运用的较为广泛，能够在长度固定且稠密的向量中对文本中的句子及单词进行充分地表达。通过在大型语料集上进行词向量的预训练，能够直接对接到下游模型处理各种任务，如机器翻译，文本分类等。词向量技术的底层对单词之间进行了距离的度量，语义相近的单词它们距离上也相应较近，它良好的表达能力能对神经网络模型的性能带来极大提升。

本文将采用 Google 的 Word2Vec 词向量技术对食谱成分数据进行预训练。本实验使用 gensim 开源组织提供的 Word2Vec 函数，它将原生 C 语言编写的 Word2Vec 通过 Python 进行了封装，并拓展出了测试功能。Word2Vec 提供了 Skip-gram 和 CBOW 两种模型，我们拟采用 CBOW 模型进行预训练，主要的参数配置见如下：

(1) train 为输入到模型的文本数据集，要求输入以空格为间隔的文本集，这里我们已经将所有食谱成分整合到了一个文档，并对它们进行了分隔。

(2) output 为输出的结果文件，将以词向量形式对结果进行保存。

(3) size 为词向量维数的设置，词向量的维数对于单词语义的表达有影响作用，在这里我们设置为默认的 100 维。

(4) window 为单词之间最大的跳跃长度，也称为训练的窗口值。这里我们使用默认值，即 window=5。设置生效后，将在训练时考虑一个单词的前 5 和后 5 的单词，而在实际运行时还有一个窗口随机选择的过程，且窗口小于等于 5。

(5) sample 设置的是单词采样的阀值。若一个单词在文档中出现的次数越多，该单词将被采样，这里我们设置为经验值 1e-5。

(6) hs 表示是否采用 Hierarchical Softmax 用于加速训练的过程，默认开启。

(7) negative 表示是否开启采样优化，我们使用默认设置。

(8) threads 用于调整并行度，结合实验机器的性能，我们设置为 12。

(9) min_count 用于设置单词过滤的词频阀值，出现频率较低的单词训练出来的向量效果不佳，因此需要进行过滤。这里我们设置为默认值 5，即过滤词频低于 5 的单词。

(10) binary 用于设置结果文件的是否以二进制进行存储，这里设置为不采用。

(11) cbow 用于设置是否采用 CBOW 模式进行训练，这里设置为 1。

本文对于词向量训练的实现见算法 4-5 所示，并列举了几个已训练好的词向量，如图 4-2 所示。

Algorithm 4-5 训练词向量

```
1.  import word2vec
2.      # 词向量预训练
3.      def init_word2vec(input, output)
4.      # 训练并保存到目标文件
5.      word2vec.word2vec(input, output, size=100, window=5, sample='1e-3', hs=1,
6.      negative=5, threads=12, min_count=5,binary=0, cbow=1)
```

本次词向量训练过后，共得到了 5697 个词向量，每个食谱成分都被表示为 100 维的词向量。图 4-2 展示了 3 个已训练好的食谱成分的词向量。

```
鲫鱼 [-0.005647, 0.021478, -0.119860, 0.047033, -0.018215, -0.044227, ...]
黄豆 [ 0.038405, 0.012784, -0.065145, 0.034277, -0.024259, -0.016884, ...]
海参 [-0.030687, 0.041931, -0.098106, 0.081450, -0.007290, -0.020126, ...]
```

图 4-2 词向量表示

4.2.2 输入数据预处理

在正式使用神经网络训练模型前，本节先对食谱样本数据进行预处理，包括食谱成分的向量化、统一输入向量的长度和标签的向量化。

(1) 食谱成分的向量化

由于上一节已经对所有食谱成分进行了词向量的训练，我们可以在之后的神经网络模型中使用这些词向量。但在此之前，我们仍需要基于已训练好的词向量的索引对食谱成分进行向量化。我们通过将样本集中的成分名与词向量文本中的进行匹配，最终将成分转换成了其词向量所在位置的索引编号。具体的实现见算法 4-6。

(2) 统一输入成分向量的长度

在第一步我们得到的成分向量长度不一致，我们需要对它们统一长度。在这里我们采取了一个策略：先计算出主料成分和辅料成分的平均长度，然后取平均长度的约 2.5 倍作为固定长度。根据这个固定长度，我们对比固定长度短的向量进行补 0，长的向量对超出部分进行截断。根据统计，我们的主料成分平均长度为 2.47，辅料成分平均长度为 5.39。因此，我们取主料成分及辅料成分向量的固定长度分别为 6、12。具体的实现见算法 4-6。

Algorithm 4-6 输入数据预处理

```
1. def data_process(corpus, csv_input, npz_output):
2.     # corpus 用于存储食谱的主辅料成分数据
3.     word2id = np.load("../datasets/ingredients_embedding1.npz")["word2id"].item()
4.     # 构成 set 用于后面的匹配
5.     word_keys = set(word2id.keys())
6.     # 成分名 → 在词向量空间对应的索引编号
7.     major = [[word2id[major if major in word_keys else '</s>'] for major in line[0]] for
       line in corpus]
8.     major_packed = pad_sequence(major, maxlen=6, padding='pre', truncating='pre',
       value=0)
9.     minor = [[word2id[minor if minor in word_keys else '</s>'] for minor in line[1]] for
       line in corpus]
10.    minor_packed = pad_sequence(minor, maxlen=12, padding='pre', truncating='pre',
       value=0)
11.    return major_packed, minor_packed
```

(3) 标签的向量化

标签的向量化有两个步骤，一是对所有标签的种类进行排序，二是将样本中的标签转成其在排序中对应的编号。具体的实现见算法 4-7。

Algorithm 4-7 标签向量化

```
1.  def flag_process(input_id_labels, outfile):
2.      all_labels = {_ for id, lables in input_id_labels for _ in lables}
3.      sorted_labels = sorted(all_labels) # 对集合里的标签进行排序
4.      label2id = {l_: ii for ii, l_ in enumerate(sorted_labels)}
5.      # 构造映射：索引 → 标签
6.      id2label = {ii: l_ for ii, l_ in enumerate(sorted_labels)}
7.      # 对样本集中的标签进行数字化
8.      d = {id: [label2id[jj] for jj in labels] for id, labels in input_id_labels}
9.      data = dict(d=d, label2id=label2id, id2label=id2label)
10.     with open(outfile, 'w+', encoding='utf-8') as f:
11.         json.dump(data, f)
12.     f.close()    # 将标签向量化后保存到 json 文件
```

4.2.3 基于 CNN 的分类模型

本节将对我们构建的卷积神经网络模型进行阐述。上文已对食谱成分长度进行了统计，输入的主料成分平均长度为 2.47，辅料成分平均长度为 5.39，合起来不超过 8，因此食谱成分数据是典型的短文本。同时，每个食谱都包含了多个标签。所以，本文基于食谱成分的多标签分类问题属于短文本的多标签分类问题。

针对这一短文本多标签分类问题，我们在 Embedding 层之后再通过不同尺寸的一维卷积核对局部特征进行提取。与原始 TextCNN[47]相比，我们对模型做了一些改进，主要表现在：将卷积层从单层拓展至双层，在卷积层和激活函数 ReLU 之间使用 BatchNorm 替代 Dropout 用于正则化，使用双层代替单层的全连接层并且加入 BatchNorm。

综上，本节所构建的卷积神经网络模型如图 4-3 所示，由下至上依次经过了输入层、Embedding 层、卷积层、正则化 BatchNorm、激活函数 ReLU、卷积层、BatchNorm 规范化、激活函数 ReLU、MaxPool 层、Concat 层。

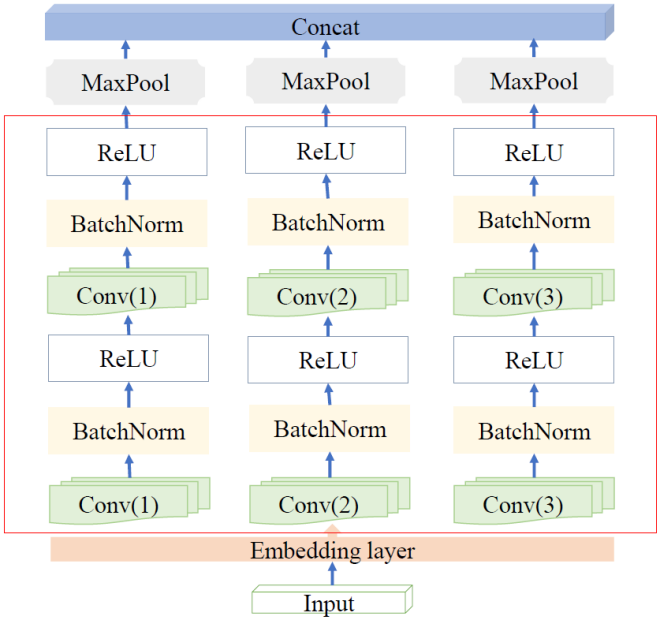

图 4-3　卷积神经网络模型结构

第一层为输入层，它主要负责将经过向量化处理的食谱成分样本输入到 CNN 模型中。

第二层为 Embedding 层，它的主要作用是将输入层传入的食谱成分所对应的数字编号转化成对应的词向量(所有成分在上文已经过了词向量的训练)。

第三层为卷积层，它的作用是提取食谱成分的局部特征，从图中可以看出它由多个不同尺度的卷积构成，即多尺度卷积层。

第四层经过了 BatchNorm，即批规范化，它能够对数据进行规范化，加速模型的收敛并且控制过拟合。

第五层为 ReLU 激活函数层，它相较于 sigmoid 和 tanh 等传统的激活函数，能更加有效地完成反向传播及梯度下降。

第六、七、八层同上，分别使用卷积层、BatchNorm、ReLU 激活函数层。

第九层为 MaxPool 层，负责从上一层中取得分数最高的特征值。

第十层为 Concat 层，负责将 MaxPool 层中得到的特征值进行拼接，最后构成一个向量。

根据上文所构建的卷积神经网络模型，我们将在本章最后的实验部分使用 Pytorch 深度学习平台完成对 CNN 的搭建。

4.2.4 基于 RNN 的分类模型

在循环神经网络模型中，我们使用双向 LSTM 对食谱短文本进行上下文特征信息的提取，它对于上下文特征信息提取的有效性已得到了验证。与常规 TextRNN[48]模型相比，我们对模型所有隐藏元进行了 K-MaxPooling 处理，而不是直接使用最后一个隐藏元作为分类。改进之后，能够利用 K-MaxPooling 的特点进而获取到更多对分类模型有效的食谱成分。

综上，本节搭建的循环神经网络的模型结构如图 4-4 所示，它由下至上依次为输入层、Embedding 层、双向 LSTM 层、K-MaxPooling 层、Concat 层、全连接层、Softmax 层。

第一层为输入层，它主要负责将经过向量化处理的食谱成分样本输入到 LSTM 模型中。

第二层为 Embedding 层，它的主要作用是将输入层传入的食谱成分所对应的数字编号转化成对应的词向量。

第三层为双向的 LSTM 层，它的主要作用是提取食谱文本的全局信息和上下文信息。

第四层为 K-MaxPooling 层，它主要作用是从上一层中获取到分数最高的 TopK 特征值。

第五层为 Concat 层，它的主要作用是将 K-MaxPooling 层中得到的特征值进行拼接，最后构成一个向量。

第六层为全连接，它负责将之前学到的所有局部特征映射到样本标记空间中。

第七层为 Softmax 层，它将多个标量映射为一个概率分布。

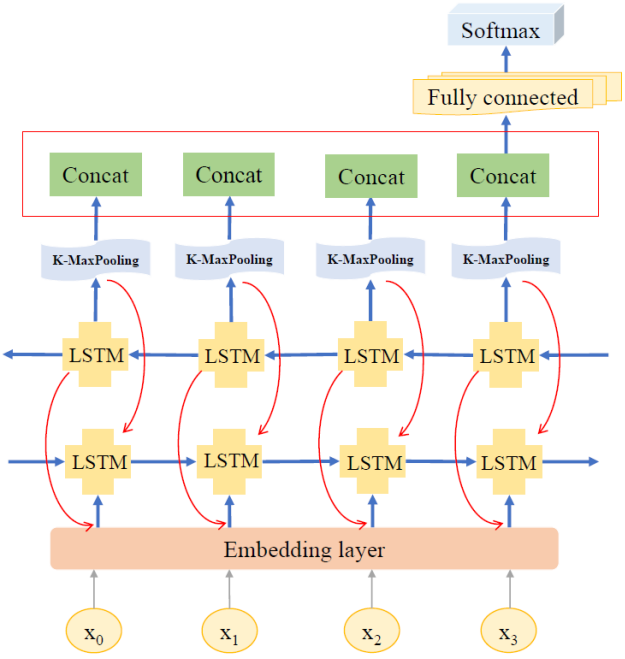

图 4-4 循环神经网络模型结构

根据上文所构建的循环神经网络模型，我们将在本章最后的实验部分使用 Pytorch 深度学习平台完成对 RNN 的搭建。

4.3 实验环境与结果分析

4.3.1 实验环境搭建

由于神经网络模型对于机器的配置要求较高，并且 GPU 在进行多维矩阵运算时速率要优于 CPU。本文将在实验中将通过 GPU 加速对模型展开训练，下面将对实验的硬件配置和软件环境进行阐述。

4.3.1.1 实验的硬件配置

为了使系统保持运行流畅，我们采用了 NVME 协议的固态硬盘和双通道的高频 DDR4 内存；为了与 CUDA 平台稳定兼容，我们采用了高功率版本的 Nvidia 显卡。具体的硬件配置如下：

(1) 处理器：Intel(R) Core(TM) i7-8750H

(2) 显卡：Nvidia GeForce GTX 1060 6GB

(3) 内存：16GB 2666Mhz DDR4

(4) 硬盘：512GB NVME SSD

4.3.1.2 实验的软件环境

随着 Facebook 公司对 Pytorch 平台的不断完善，自 1.0 版本开始，就对 Windows 系统进行了兼容支持。为了方便实验的开展，我们采用的深度学习平台是带 GPU 支持的 Pytorch 1.3，操作系统为 Window 10，开发语言为 Python 3.6，机器学习平台采用的是 Scikit-Learn 0.19.1。

我们使用发行版的 Anaconda 部署 Python 的语言环境。Anaconda 是一个集成了 Python 运行环境的开源工具包，它能够协调好各个组件的版本兼容性，并且附带了 Scikit-Learn 机器学习框架。

通过在 Nvidia 官网上查阅关于 CUDA 支持的说明文档，了解到本实验所用显卡支持 CUDA 9.2 版本。因此，本文在实验机器上安装了 CUDA 9.2 环境用于支持后期训练模型的 GPU 加速。

为了配合 CUDA 9.2 和 Python 3.6 的使用，我们查阅了 Pytorch 官网，制定了适合的部署方案，具体如图 4-5 所示。我们在系统命令行上使用命令“conda install pytorch torchvision cudatoolkit=9.2 -c pytorch -c defaults -c numba/label/dev”便可以对定制的方案进行部署。

到此，我们完成本文实验中的环境搭建。

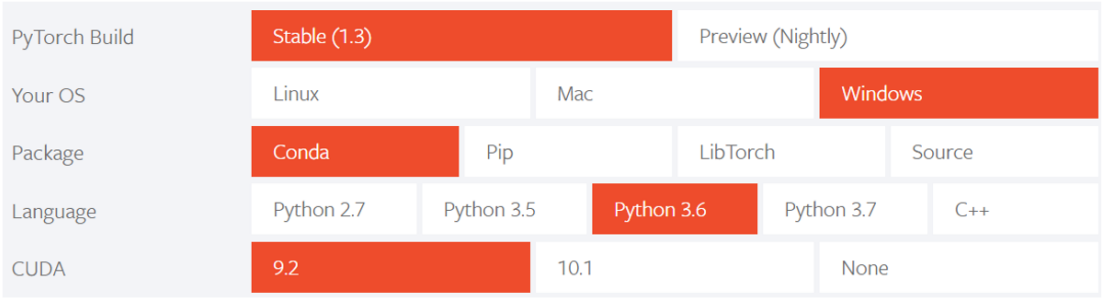

图 4-5 深度学习环境搭建

4.3.2 实验过程和结果分析

本节主要对上文所构建的模型进行训练以及参数调优，同时对于测试样本进行测试，并对测试结果进行分析。接着，我们将选择出测试分数最好的模型视作最终的多标签分类模型。

本次实验所用到的食谱样本共计 15,747 条，标签共 447 种，我们对样本的划分如下：训练集为 12,747 条，测试集为 3,000 条，并从训练集中取 10%作为验证集。同时，为了扩充样本数，我们还在模型训练当中引入了数据增强策略。

下面我们先给出分类模型的评价指标，接着通过卷积神经网络和循环神经网

络对模型进行训练，并通过大量的实验对模型的参数进行调优，得出效果更优的模型。

4.3.2.1 模型性能评价指标

本小节将对本文基于食谱成分的多标签分类模型的性能评价指标进行说明，这里我们采用 ROC-AUC 值和汉明损失(Hamming_loss)作为模型的评价指标，下面将分别对它们进行阐述。

(1) ROC-AUC 值

ROC-AUC 所表达的是：ROC(Receiver Operating Characteristic)曲线下的面积 AUC(Area Under ROC)。首先，ROC 曲线是在设定了不同的阀值下由 FPR 作为横坐标和 TPR 作为纵坐标所绘制的，它直观地反映了分类模型的泛化性能好坏。

$$TPR = \frac{TP}{TP+FN} \quad \text{公式(4-1)}$$

$$FPR = \frac{FP}{TN+FP} \quad \text{公式(4-2)}$$

由于 ROC 曲线可能会遇上交叉的情况，为了方便对模型进行性能评估，通常需要引入 AUC 面积，它反映了所预测到的正例排在负例之前的概率，并且能够有效地对分类性能进行评估。ROC-AUC 值越接近 1 代表分类效果越好，一个简单的计算用例见图 4-6。

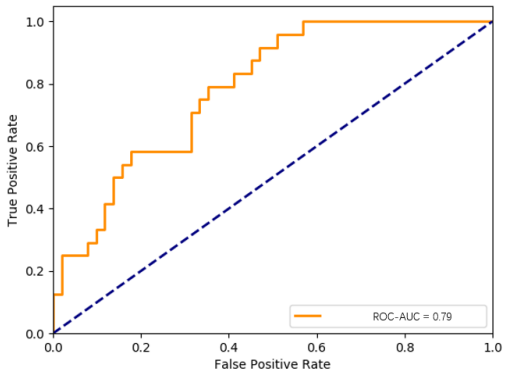

图 4-6 ROC-AUC 计算用例

(2) 汉明损失

汉明损失是错误预测的标签的分数，它表示了所有标签中错误样本所占的比例，因此汉明损失值越小则代表模型的分类性能更优。汉明损失的具体计算见公式(4-3)。

$$HammingLoss(x_i, y_i) = \frac{1}{|D|}\sum_{i=1}^{|D|}\frac{xor(x_i, y_i)}{|L|} \quad \text{公式(4-3)}$$

其中，$|D|$为所有样本的总个数，$|L|$为所有标签的总个数，x_i与y_i分别为预测的结果和真实结果，xor为异或运算。

4.3.2.2 卷积神经网络的实验

本小节将对卷积神经网络模型进行训练，涉及调优的参数有：多尺度卷积核、卷积核数目、是否数据增强。

(1) 多尺度卷积核

本节我们设置了多种尺度的卷积核，并在训练集上对卷积神经网络模型进训练。通过对得到的模型在测试集中进行评估，旨在探索多尺度的卷积核对模型分类效果的影响。

卷积核尺度的预设值分别为 1、2、3、1-3、1-2-3，我们每次训练卷积神经网络模型之前，都只单独对卷积核尺度进行调整。在测试阶段，我们采用 ROC-AUC 值和 HammingLoss 进行模型的性能评估。实验结果如表 4-1 所示。

表 4-1 调整卷积核尺度的实验结果

编号	卷积核尺度	ROC-AUC	HammingLoss
1	window_sizes = 1	98.07%	0.0126
2	window_sizes = 2	98.21%	0.0124
3	window_sizes = 3	97.60%	0.0137
4	window_sizes = 1-3	98.34%	0.0122
5	**window_sizes = 1-2-3**	**98.39%**	**0.0120**

从表中可以看出，当卷积核尺度为 1-2-3 时，卷积神经网络在食谱成分的多标签分类中效果最好。并且单个卷积核的分类效果都比多尺度卷积核的要差，说明多种尺度的卷积核有更好的感受野，能捕捉到食谱文本更多的特征信息。

(2) 卷积核数目

本小节通过设定不同的卷积核数对卷积神经网络模型进行训练，然后通过测试集进行性能评估。

我们保持卷积核尺度为 1-2-3 的前提下，对卷积核数分别设定为 64、100、156、200、250、300。实验结果如表 4-2 所示。

表 4-2 调整卷积核数的实验结果

编号	卷积核数	ROC-AUC	HammingLoss
1	64	98.26%	0.0123
2	100	98.04%	0.0128
3	156	98.25%	0.0123
4	200	98.34%	0.0121
5	**250**	**98.39%**	**0.0120**
6	300	98.26%	0.0122

从上表可以看出，当卷积核的数量为 250 时，卷积神经网络的分类效果最佳，但与卷积核的数量为 200 时效果相差不大。随着卷积核的数目增加至 300 时，其分类效果甚至与卷积核数为 156 时相近。说明盲目增加卷积核的数目并不一定能带来更好的分类效果，同时卷积核数的增多还会降低训练的速度。

(3) 是否数据增强

本小节在卷积核尺度为 1-2-3、卷积核数为 250 的前提下对是否在卷积神经网络的训练中加入数据增强进行了实验。关于数据增强的具体内容我们已在 2.3.3 进行了详细的介绍，此处不再阐述。实验结果如表 4-3 所示。

表 4-3 调整数据增强的实验结果

编号	是否数据增强	ROC-AUC	HammingLoss
1	否	98.39%	0.0120
2	**是**	**98.40%**	**0.0119**

通过表的实验结果我们可以看出，数据增强策略的引入，对于卷积神经网络模型的分类效果有所提升，但对本文基于食谱成分的多标签分类任务提升不是特别明显。

综上实验结果分析，我们可以看出当卷积神经网络模型的卷积尺度设定为 1-2-3，卷积核数设为 250 以及选择数据增强，此时模型的分类效果最佳，ROC-AUC 值能达到 98.40%，汉明损失为 0.0119。

4.3.2.3 循环神经网络的实验

本节将对循环神经网络模型进行训练，涉及调优的参数有：单双向 LSTM 选择及池化方式、隐藏状态参数、是否数据增强。

(1) 单双向 LSTM 及池化方式

本小节我们将设定不同的池化方式，依次在选择单向或双向 LSTM 的循环神经网络模型中进行训练。实验结果见表 4-4。

表 4-4 调整单双向 LSTM 和池化方式的实验结果

编号	单/双向 LSTM	池化方式	ROC-AUC	HammingLoss
1	单	2-MaxPooling	97.13%	0.0148
2	**双**	**2-MaxPooling**	**98.18%**	**0.0125**
3	单	3-MaxPooling	97.54%	0.0141
4	**双**	**3-MaxPooling**	**97.97%**	**0.0129**
5	单	4-MaxPooling	97.37%	0.0143
6	**双**	**4-MaxPooling**	**98.03%**	**0.0128**

通过对表中实验结果的分析可以看出，选择双向 LSTM 的循环神经网络在本文基于食谱成分的多标签分类任务中表现都要优于同条件下的单向 LSTM 模型。这表明双向 LSTM 的循环神经网络模型对于特征信息的提取要优于单向 LSTM，并且当池化方式设定为 2-MaxPooling 时，模型的分类效果最好。

(2) 隐藏状态参数

本小节我们将对循环神经网络模型设置不同的隐藏状态参数，并在食谱的训练样本中进行模型训练。本次实验在模型设定为双向 LSTM 且选用 2-MaxPooling 池化方式的前提下，拟设置 5 个不同隐藏状态参数，分别为 64、128、200、256、320。实验结果见表 4-5。

表 4-5 调整隐藏状态参数的实验结果

编号	隐藏状态参数	ROC-AUC	HammingLoss
1	64	97.83%	0.0133
2	128	98.14%	0.0126
3	**200**	**98.18%**	**0.0125**
4	256	97.67%	0.0135
5	320	98.08%	0.0127

通过表格的实验结果我们可以看到，当隐藏状态参数设定为 200 时循环神经网络模型的分类效果最优，当设为 256 时分类效果最差。当隐藏状态设置地过大时，并没有对模型的分类效果有很大的提升，反而降低了模型训练的速度。

(3) 是否数据增强

在循环神经网络模型设定为双向 LSTM 且选用 2-MaxPooling 池化方式的基础上，对是否在训练模型时引入数据增强进行了实验。实验结果见表 4-6。

表 4-6 调整数据增强的实验结果

编号	是否数据增强	ROC-AUC	HammingLoss
1	否	98.18%	0.0125
2	**是**	**98.25%**	**0.0122**

从表中的数据可以看出，引入数据增强策略对模型的分类效果有一定的提升作用。综上实验结果可以得出，当循环神经网络模型设定为双向 LSTM、2-MaxPooling 池化方式以及采用数据增强时，模型在本文的基于食谱成分的多标签分类任务中表现效果最好，ROC-AUC 值能达到 98.25%，汉明损失降至 0.0122。

4.3.3 实验结果对比

为了与本文构建的卷积神经网络以及循环神经网络进行性能对比，本节在基于食谱成分的多标签分类任务中使用了传统的机器学习算法，它们分别是贝叶斯模型、K 近邻模型，具体的实现已经在 4.1 进行了阐述。我们在食谱成分样本集中对机器学习模型进行训练，接着对测试集进行了测试。实验结果见表 4-7。

表 4-7 各分类模型的性能对比结果

编号	模型	ROC-AUC	HammingLoss
1	贝叶斯	79.66%	0.1141
2	K 近邻	93.97%	0.0158
3	**CNN**	**98.40%**	**0.0119**
4	RNN	98.25%	0.0122

通过分析表的实验结果可以得出，卷积神经网络模型在本文的基于食谱成分多标签的分类任务中表现最出色，ROC-AUC 值达到了 98.40%，其次是循环神经网络模型。同时，神经网络模型的分类效果都要优于传统的贝叶斯和 K 近邻模型。我们另外还发现，通过直接改编算法实现多标签分类的 K 近邻模型的分类效果要比使用二元关联法的贝叶斯模型要好得多，可能的原因是二元关联法丢失了较多标签之间的信息，从而影响了最终的分类效果。

4.4 本章小结

本章在基于食谱成分的多标签分类任务中，给出了传统机器学习算法中贝叶斯和 K 近邻分类算法的实现，同时在基于 Pytorch 深度学习平台下完成了卷积神经网络及循环神经网络的模型实现。最后，我们通过大量的实验对神经网络模型

进行了参数调优，接着选出分类效果最优的神经网络模型，并与传统机器学习模型进行了分类效果的对比。实验结果证明，数据增强对于神经网络模型的分类效果有一定的提升，同时神经网络模型在本文的食谱多标签分类任务中的效果要优于一些传统机器学习模型。

第五章 总结与展望

在本章，我们将对本文的研究工作进行总结，并对将来进一步的研究内容进行探索与展望。

5.1 本文总结

随着在线食谱社区的不断发展，人们对于食谱的分享由线下拓展至了线上。日益增多的线上用户为在线食谱社区的建设带来了更多丰富的内容，也为本文关于中国在线食谱的研究带来了丰富的数据资源。

本文首先通过数据采集技术获取到了足量的中国食谱数据，在对数据完成了清洗工作后，运用一些数据统计和挖掘的技术对中国在线食谱进行了探索式地数据分析，并对相应研究结果进行了可视化。主要完成的内容包括对中国在线食谱的成分多样性分析、特色成分分析、复杂性分析、相似性分析以及惯用辅料成分的分析。我们的研究主要得出了如下结论：成分的消费多样性和组合多样性有较高的关联性，一些存在于特定区域的食谱成分可以作为显著的菜系代表，地理位置对食谱之间的相似性存在一定的影响。我们还对中国各大菜系的食谱复杂性进行了评估，通过运用频繁项集挖掘算法，我们还找出了惯用的辅料成分组合。

互联网对于标签的运用越来越广泛，很多社交软件都建立它们各自的标签体系。对于本文，我们基于在线食谱成分完成了多标签文本分类的任务，主要用到了卷积神经网络和循环神经网络。在对神经网络模型进行训练的过程中，我们通过了大量的实验进行了模型参数的调优，并且通过贝叶斯和 K 近邻算法作为基准对各个模型的分类效果进行了对比。实验表明，基于神经网络的多标签分类模型在本文的任务中均达到了较好的分类效果，并优于传统机器学习模型。

5.2 工作展望

本文的研究工作还存在一些不足之处，将来可以从如下方面进行完善：

(1) 进一步丰富食谱数据样本，通过更多的食谱成分数据集来进行词向量的训练，从而学习到更好的文本语义表达空间。

(2) 通过研究在线食谱网站的用户对于食谱标签的选择偏好，尝试设计一种基于标签的食谱推荐功能，以匹配用户的需求。

参考文献

[1] Ma, Guansheng. Food, eating behavior, and culture in Chinese society[J]. Journal of Ethnic Foods, 2015, 2(4): 195-199.

[2] D. Tilman and M. Clark. Global diets link environmental sustainability and human health[J]. Nature, 2014, 515(7528): 518-522.

[3] Ahn Y Y, Ahnert S E, Bagrow J P, et al. Flavor network and the principles of food pairing[J]. Scientific reports, 2011, 1: 196.

[4] Teng C Y, Lin Y R, Adamic L A. Recipe recommendation using ingredient networks[C]//Proceedings of the 4th Annual ACM Web Science Conference. 2012: 298-307.

[5] West R, White R W, Horvitz E. From cookies to cooks: Insights on dietary patterns via analysis of web usage logs[C]//Proceedings of the 22nd international conference on World Wide Web. 2013: 1399-1410.

[6] Wagner C, Singer P, Strohmaier M. The nature and evolution of online food preferences[J]. EPJ Data Science, 2014, 3: 1-22.

[7] Said A, Bellogín A. You are What You Eat! Tracking Health Through Recipe Interactions[C]//Rsweb@ recsys. 2014.

[8] Abbar S, Mejova Y, Weber I. You tweet what you eat: Studying food consumption through twitter[C]//Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems. 2015: 3197-3206.

[9] Kusmierczyk T, Trattner C, Nørvåg K. Temporality in online food recipe consumption and production[C]//Proceedings of the 24th International Conference on World Wide Web. 2015: 55-56.

[10] Müller M, Harvey M, Elsweiler D, et al. Ingredient matching to determine the nutritional properties of internet-sourced recipes[C]//2012 6th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth) and Workshops. IEEE, 2012: 73-80.

[11] De Choudhury M, Sharma S, Kiciman E. Characterizing dietary choices, nutrition, and language in food deserts via social media[C]//Proceedings of the 19th acm conference on computer-supported cooperative work & social computing. 2016: 1157-1170.

[12] Min W, Bao B K, Mei S, et al. You are what you eat: Exploring rich recipe information for cross-region food analysis[J]. IEEE Transactions on Multimedia, 2017, 20(4): 950-964.

[13] Nedovic V. Learning recipe ingredient space using generative probabilistic models[C]//Proceedings of Cooking with Computers Workshop (CwC). 2013, 1:

13-18.

[14] Zhu Y X, Huang J, Zhang Z K, et al. Geography and similarity of regional cuisines in China[J]. PloS one, 2013, 8(11).

[15] Read J, Pfahringer B, Holmes G, et al. Classifier chains for multi-label classification[J]. Machine learning, 2011, 85(3): 333.

[16] Tsoumakas G, Katakis I, Vlahavas I. Random k-labelsets for multilabel classification[J]. IEEE Transactions on Knowledge and Data Engineering, 2010, 23(7): 1079-1089.

[17] Cheng W, Hüllermeier E, Dembczynski K J. Bayes optimal multilabel classification via probabilistic classifier chains[C]//Proceedings of the 27th international conference on machine learning (ICML-10). 2010: 279-286.

[18] K. Dembczynski, W. Waegeman, W. Cheng, et al. On label dependence in multi-label classification[J]. Machine Learning, 2012, 88(1-2): 5-45.

[19] Ji S, Tang L, Yu S, et al. A shared-subspace learning framework for multi-label classification[J]. ACM Transactions on Knowledge Discovery from Data (TKDD), 2010, 4(2): 1-29.

[20] Xia X, Yang X, Li S, et al. RW. KNN: A proposed random walk KNN algorithm for multi-label classification[C]//Proceedings of the 4th workshop on Workshop for Ph. D. students in information & knowledge management. 2011: 87-90.

[21] Zhang M L, Wu L. Lift: Multi-label learning with label-specific features[J]. IEEE transactions on pattern analysis and machine intelligence, 2014, 37(1): 107-120.

[22] Zhang M L, Zhang K. Multi-label learning by exploiting label dependency[C]//Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining. 2010: 999-1008.

[23] Wang H, Ding C, Huang H. Multi-label classification: Inconsistency and class balanced k-nearest neighbor[C]//Twenty-Fourth AAAI Conference on Artificial Intelligence. 2010.

[24] Kandola E. J., Hofmann T., Poggio T., et al. A Neural Probabilistic Language Model[J]. Studies in Fuzziness and Soft Computing, 2006, 194: 137-186.

[25] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov. Bag of tricks for efficient text classification[C]//Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, 2017: 427–431.

[26] Kalchbrenner, E. Grefenstette, P. Blunsom. A Convolutional Neural Network for Modelling Sentences[C]//Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, 2014, 1: 655-665.

[27] Liu P, Qiu X, Huang X. Recurrent neural network for text classification with multi-task learning[C]//Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, 2016: 2873-2879.

[28] Lai S, Xu L, Liu K, et al. Recurrent convolutional neural networks for text classification[C]//Twenty-ninth AAAI conference on artificial intelligence. 2015.

[29] 孙立伟, 何国辉, 吴礼发. 网络爬虫技术的研究[J]. 电脑知识与技术, 2010, 6(15): 4112-4115.

[30] 王芳, 张睿, 宫海瑞. 基于 Scrapy 框架的分布式爬虫设计与实现[J]. 信息技术, 2019(03): 96-101.

[31] C. E. Shannon. A mathematical theory of communication[J]. Mobile Computing and Communications Review, 2001, 5(1): 3-55.

[32] 张保富, 施化吉, 马素琴. 基于 TFIDF 文本特征加权方法的改进研究[J]. 计算机应用与软件, 2011, 28(02): 17-20.

[33] 陈二静, 姜恩波. 文本相似度计算方法研究综述[J]. 数据分析与知识发现, 2017, 1(06): 1-11.

[34] Y. Djenouri, D. Djenouri, J. C.-W. Lin, and A. Belhadi. Frequent Itemset Mining in Big Data With Effective Single Scan Algorithms[J]. IEEE Access, 2018, 6: 68013-68026.

[35] 饶正婵, 范年柏. 关联规则挖掘 Apriori 算法研究综述[J]. 计算机时代, 2012(09): 11-13.

[36] 何金凤. 基于中文信息检索的文本预处理研究[D]. 成都：电子科技大学, 2008.

[37] 彭君睿. 面向文本分类的特征提取算法研究[D]. 北京：北京邮电大学, 2014.

[38] J. W. Wei and K. Zou. EDA: easy data augmentation techniques for boosting performance on text classification tasks[C]//Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, 2019: 6381-6387.

[39] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space[C]//1st International Conference on Learning Representations, 2013.

[40] Mikolov T, Sutskever I, Chen K, et al. Distributed representations of words and phrases and their compositionality[C]//Advances in neural information processing systems. 2013: 3111-3119.

[41] 阿曼. 朴素贝叶斯分类算法的研究与应用[D]. 大连：大连理工大学, 2014.

[42] 桑应宾. 基于 K 近邻的分类算法研究[D]. 重庆：重庆大学, 2009.

[43] 刘建伟, 刘媛, 罗雄麟. 深度学习研究进展[J]. 计算机应用研究, 2014, 31(07): 1921-1930+1942.

[44] 李飞腾. 卷积神经网络及其应用[D]. 大连：大连理工大学, 2014.

[45] 杨丽, 吴雨茜, 王俊丽, 刘义理. 循环神经网络研究综述[J]. 计算机应用, 2018, 38(S2): 1-6+26.

[46] Waltman L, Van Eck N J, Noyons E C M. A unified approach to mapping and clustering of bibliometric networks[J]. Journal of Informetrics, 2010, 4(4): 629-635.

[47] Kim Y. Convolutional neural networks for sentence classification[J]. arXiv preprint arXiv:1408.5882, 2014.

[48] Peng J, Huang Z, Cheng J. A Deep Recurrent Network for web server performance prediction[C]//2017 IEEE Second International Conference on Data Science in Cyberspace (DSC). IEEE, 2017: 500-504.

发表论文和参加科研情况

[1] Chuitian Rong, Zhaopei Liu, Na Huo and Huabo Sun. Exploring Chinese Dietary Habits Using Recipes Extracted From Websites[J]. IEEE Access, 2019, 7: 24354-24361.

致谢

时光匆匆，转眼间研究生的学习阶段即将步入尾声。在天津工业大学这两年半的学习时光里，虽然充满了挑战和艰辛，但奋斗的时光总是充实的。

首先我要感谢的是荣垂田老师，他是我研究生期间的引路人。他总是以身作则，对待科研工作总是严谨细致，对于细节的把握有着更高的要求。在科研工作中，他能够与学生进行积极的探讨，聆听学生的汇报意见。同时，他突出的学术能力以及耐心的处事风格，都让我受益匪浅。他能够结合学生的特点制定合理的培养计划，这让学生的综合能力得到了进一步的提升。不管是在研究方向的选择、研究工作的开展、还是论文的写作，他总能给予我耐心的指导，授之以渔。在生活上，他同样给予了学生许多的关怀，他用自己的言行鼓励学生积极面对生活中的困难。荣老师教会我的不仅是严谨细致的科研工作态度，更重要的是，是他那严以律己宽以待人的处事风格。我会努力向他学习，将这些优秀的品质运用到今后的工作和学习中。衷心祝愿荣老师工作顺利，身体安康！

其次，我要感谢研究生期间给予我帮助和支持的各位老师、同学、师兄师弟师妹，还有朝夕相伴的舍友。你们总能够在我遇到困惑时，给予我耐心解答；总能在我受挫时，给予信心和鼓励。我十分珍惜与你们在学校一同成长和学习的时光，这将会是我人生中难忘的记忆。在这里，由衷地祝福你们！在青春岁月里，感谢有你们！

感谢母校天津工业大学对我的培养，为我提供良好的工作和学习环境。天津工业大学留下了我人生中许多美好的记忆，对于即将毕业的我，对母校十分不舍。今后，我将继续努力奋斗，以实际行动维护母校的荣誉，并为母校贡献出应有的力量。衷心祝愿我的母校，天津工业大学继续迈向新的辉煌！

最后，我向百忙之中评阅论文和负责答辩的各位专家、教授致以最真挚的敬意！