



文本分类中的特征选择

 baiziyu
安心记录每一刻

关注他

8 人赞同了该文章

特征选择在文本分类中指的是筛选从训练集中得到的词汇表中的词语。去掉那些对类别没有指征作用的词。比如某个词语很少出现，却又都集中出现在某一个类别中，并且与类别又没有什么关系，对于这样的词语就应当从词汇表中去除。去除一些没有必要的词语，使得特征向量的维度降低，不仅可以提高模型的训练和预测时间，而且在某些模型中可以提高预测精度。针对某一个具体类别的**标准特征选择流程**如下：

输入：文档集，类别名称，需要抽取出的特征词数量

输出：最佳特征词列表 L'

步骤：

- (1) 扫描文档集得到词汇表 V
- (2) 初始化候选特征词列表 L
- (3) 从词汇表 V 中读取一个词语 t
- (4) 计算效用指标 $A(t,c)$

Medical 治疗 368.0099534384237 疗效 248.5683626864011 患者 241.35098664035846 病人 169.57704337451065 肿瘤 148.35584261922057 医院 143.7391567175393 癌症 142.88300058328016 手术 136.34570477247502 外科 125.96090451366103 临床 121.06540837233051	Sports 农业 1435.160285416867 农产品 797.438645422512 产量 652.6958119639446 粮食 638.809265570511 训练 597.2331401001437 农村 588.0870413378718 比赛 578.6929977131628 农户 573.7547140476672 土地 545.0782615025894 耕地 506.81372161370456	Agriculture 农业 1917.364197004155 农产品 1045.8375308438606 粮食 836.8183024183533 产量 820.2077664482362 农村 797.9503296860242 农户 755.201145705948 土地 733.2857160468188 耕地 666.7895067793634 面积 610.7652521669077 品种 590.2273092862546
Education 六一 67.49722950858728 今天下午 52.76692698902033 后天下之乐而乐 52.45705681207393 公而忘私 52.45705681207393 阿姨 52.45705681207393 拼命 52.45705681207393 无政府主义 52.45705681207393 办得更好 52.45705681207393 教育电视台 52.45705681207393 倡议书 52.45705681207393	Electronics 集成电路 403.36287092958247 半导体 399.8186372766048 电子产品 399.8186372766048 电子部 284.4972962524497 电子 240.2675387388617 工业园区 199.58052041685784 电子元件 199.58052041685784 科研开发 131.17323206699157 录像机 125.30261148932676 2 7 0 0 万 117.77841292622638	Communication 邮电部 686.5554648733478 通信网 599.2090220100024 通信 590.1858207617084 邮电 394.13444597832074 卫星通信 255.1420080188027 光缆 237.09923663286878 移动电话 215.84702089109126 交换机 215.84702089109126 电信 180.3210449149559 覆盖全国 142.02536989415796

频率

频率计算方法有两种，第1种是词项t在c类文档集中出现次数/c类文档集包含的词项总数。第2种是c类文档集中包含词项t的文档数/c类文档集文档总数。做文本分类的人应该注意到频率的特征抽取也就是根据TF值的特征抽取，它跟文本的向量表示是一回事，不依赖于文本的类别标注，但是TF-IDF值是依赖文本类别标记的。

例子：在含有6个类别的文档集中进行频率特征抽取的例子

Medical 治疗 0.5098039215686274 专家 0.3333333333333333 医疗 0.29411764705882354 医院 0.27450980392156865 患者 0.27450980392156865 发现 0.2549019607843137 百分之 0.23529411764705882 临床 0.21568627450980393 疗效 0.21568627450980393 疾病 0.19607843137254902	Sports 文献 0.7086991221069433 中的 0.6065442936951316 原刊期 0.6065442936951316 原刊页 0.6033519553072626 培养 0.4764565043894653 学习 0.4533120510774142 训练 0.4301675977653631 第一 0.4205905826017558 重视 0.4158020750199521 学校 0.4142059058260176	Agriculture 农业 0.9520078354554359 农村 0.66307541625857 文献 0.6258570029382958 中的 0.6199804113614104 农产品 0.5690499510284035 原刊期 0.5582761998041136 原刊页 0.5582761998041136 作者 0.5288932419196866 土地 0.5004897159647405 措施 0.49167482859941236
Education 学校 0.6440677966101694 学生 0.4915254237288136 培养 0.4406779661016949 学习 0.423728813559322 教师 0.423728813559322 北京 0.3389830508474576 文化 0.288135593220339 知识 0.2711864406779661 事业 0.2711864406779661 教学 0.2711864406779661	Electronics 电子 0.6296296296296297 计算机 0.4074074074074074 美元 0.37037037037037035 设备 0.2962962962962963 集成电路 0.2962962962962963 工业 0.2962962962962963 1 0 0.25925925925925924 一家 0.25925925925925924 芯片 0.25925925925925924 本报 0.25925925925925924	Communication 通信 0.72 邮电 0.4 通信网 0.4 电话 0.36 北京 0.36 设备 0.32 业务 0.32 邮电部 0.32 中心 0.28 上海 0.28

从结果来看，卡方和互信息抽取出的特征词是差不多的。并且“无政府主义”等长词肯定是只出现了很少的次数，也就是说卡方和互信息对于罕见词是很敏感的，但是这样的词从语义上跟类别“教育”并没有直接关系；相反我们可以看到在频率选出的“教育”类词汇中，没有这样的罕见词。在实际应用中，我们可以取卡方（或互信息）与频率两种方法抽取出的特征词的交集作为词汇表。另外，我们也可以通过人工挑选出对于类别识别没有意义的词语作为停用词比如数词“10”，某些名词“原刊期”，“原刊页”等。特别注意到“运动”类中一定掺有了“农业”类的文本，这样的文本的过滤我们可以通过介绍了短语特征抽取后，利用短语特征予以滤除。也可以将“农业”类的文本聚类，聚类后抽取每个簇的特征词进而滤除非“农业”类的文本，关于聚类我们在后边再做介绍。写到这儿，才想起来忘记先介绍HanLP中的分词了，我们将在明天先介绍HanLP分词的内容。^_^。

三种特征抽取方法的示例代码在

```
#coding:utf-8
'''
互信息、卡方、频率征选择示例
'''
```



```

from math import log2
from sklearn.datasets import load_files
from pyhanlp import *

# 加载实词分词器 参考https://github.com/hankcs/pyhanlp/blob/master/tests/demos/demo\_noti
Term = JClass("com.hankcs.hanlp.seg.common.Term")
NotionalTokenizer = JClass("com.hankcs.hanlp.tokenizer.NotionalTokenizer")

def getDocuments(root_path, file_path_li, file_encoding="utf-8"):
    """
    读取原始文档集并进行预处理
    :param file_path: 文档集所在路径
    :return: 预处理后的文档列表
    """
    all_text = []
    all_data = load_files(container_path=root_path, categories=file_path_li,
                          encoding=file_encoding, decode_error="ignore")
    for label, raw_text in zip(all_data.target, all_data.data):
        word_li = preprocess(raw_text)
        label = all_data.target_names[label]
        all_text.append((label, set(word_li)))
    return all_text

def preprocess(raw_text):
    """
    预处理
    :param raw_text:
    :return:
    """
    # 将换行回车符替换为空格
    raw_text = re.sub(u'\r|\n', ' ', raw_text)
    # 去掉数值字母
    raw_text = re.sub(u'[0-9a-zA-Z\.\.]+', u'', raw_text)
    # 分词
    word_li = [w.word for w in NotionalTokenizer.segment(raw_text)]
    # 去除空白符
    word_li = [w.strip() for w in word_li if w.strip()]
    # 移除单字词
    word_li = [w for w in word_li if len(w)>1]
    return word_li

def getVocabulary(all_text):
    """
    获取文档集词汇表
    :param all_text:
    :return:
    """
    global vocabulary
    for label, word_set in all_text:
        vocabulary |= word_set

def mutual_infomation(N_10, N_11, N_00, N_01):
    """
    互信息计算
    :param N_10:
    :param N_11:
    :param N_00:
    :param N_01:
    :return: 词项t互信息值
    """

```



```

I_UC = (N_11 * 1.0 / N) * log2((N_11 * N * 1.0) / ((N_11 + N_10) * (N_11 + N_01)))
        (N_01 * 1.0 / N) * log2((N_01 * N * 1.0) / ((N_01 + N_00) * (N_01 + N_11)))
        (N_10 * 1.0 / N) * log2((N_10 * N * 1.0) / ((N_10 + N_11) * (N_10 + N_00)))
        (N_00 * 1.0 / N) * log2((N_00 * N * 1.0) / ((N_00 + N_10) * (N_00 + N_01)))
return I_UC

def chi_square(N_10, N_11, N_00, N_01):
    """
    卡方计算
    :param N_10:
    :param N_11:
    :param N_00:
    :param N_01:
    :return: 词项t卡方值
    """
    fenzi = (N_11 + N_10 + N_01 + N_00)*(N_11*N_00-N_10*N_01)*(N_11*N_00-N_10*N_01)
    fenmu = (N_11+N_01)*(N_11+N_10)*(N_10+N_00)*(N_01+N_00)
    if fenmu == 0:
        return 0
    return fenzi*1.0/fenmu

def freq_select(t_doc_cnt, doc_cnt):
    """
    频率特征计算
    :param t_doc_cnt: 类别c中含有词项t的文档数
    :param doc_cnt: 类别c中文档总数
    :return: 词项t频率特征值
    """
    return t_doc_cnt*1.0/doc_cnt

def selectFeatures(documents, category_name, top_k, select_type="chi"):
    """
    特征抽取
    :param documents: 预处理后的文档集
    :param category_name: 类目名称
    :param top_k: 返回的最佳特征数量
    :param select_type: 特征选择的方法, 可取值chi,mi,freq, 默认为chi
    :return: 最佳特征词序列
    """
    L = []
    # 互信息和卡方特征抽取方法
    if select_type == "chi" or select_type == "mi":
        for t in vocabulary:
            N_11 = 0
            N_10 = 0
            N_01 = 0
            N_00 = 0
            N = 0
            for label, word_set in documents:
                if (t in word_set) and (category_name == label):
                    N_11 += 1
                elif (t in word_set) and (category_name != label):
                    N_10 += 1
                elif (t not in word_set) and (category_name == label):
                    N_01 += 1
                elif (t not in word_set) and (category_name != label):
                    N_00 += 1
            else:
                print("N error")
                exit(1)

```



```

        continue
    # 互信息计算
    if select_type == "mi":
        A_tc = mutual_infomation(N_10, N_11, N_00, N_01)
    # 卡方计算
    else:
        A_tc = chi_square(N_10, N_11, N_00, N_01)
    L.append((t, A_tc))
# 频率特征抽取法
elif select_type == "freq":
    for t in vocabulary:
        # C类文档集中包含的文档总数
        doc_cnt = 0
        # C类文档集包含词项t的文档数
        t_doc_cnt = 0
        for label, word_set in documents:
            if category_name == label:
                doc_cnt += 1
                if t in word_set:
                    t_doc_cnt += 1
        A_tc = freq_select(t_doc_cnt, doc_cnt)
        L.append((t, A_tc))
    else:
        print("error param select_type")
    return sorted(L, key=lambda x:x[1], reverse=True)[:top_k]

# 定义词汇表
vocabulary = set()

def main():
    # 读取文档集（需要根据具体类目名称修改）
    category_name_li = ["Medical", "Sports", "Agriculture",
                        "Education", "Electronics", "Communication"]
    # 获取文本（根目录需要根据具体类目名称修改）
    all_text = getDocuments(r"data/news", category_name_li, "gbk")
    print("all_text len = ", len(all_text))
    # 读取词汇表
    getVocabulary(all_text)
    print("vocabulary len = ", len(vocabulary))
    # 获取特征词表
    print("=*20, '\n', " 卡方特征选择  \n", "=*20)
    feature_select_type = "chi"
    for category_name in category_name_li:
        # 特征抽取，最后一个参数可选值 "chi"卡方, "mi"互信息, "freq"频率
        feature_li = selectFeatures(all_text, category_name, 10, feature_select_type)
        print(category_name)
        for t, i_uc in feature_li:
            print("%s\t%.3f" % (t, i_uc))
        print("=*10)

    print("=*20, '\n', " 互信息特征选择  \n", "=*20)
    feature_select_type = "mi"
    for category_name in category_name_li:
        # 特征抽取，最后一个参数可选值 "chi"卡方, "mi"互信息, "freq"频率
        feature_li = selectFeatures(all_text, category_name, 10, feature_select_type)
        print(category_name)
        for t, i_uc in feature_li:
            print("%s\t%.3f" % (t, i_uc))
        print("=*10)

    print("=*20, '\n', " 频率特征选择  \n", "=*20)

```

```
# 特征抽取, 最后一个参数可选值 "chi" 卡方, "mi" 互信息, "freq" 频率
feature_li = selectFeatures(all_text, category_name, 10, feature_select_type)
print(category_name)
for t, i_uc in feature_li:
    print("%s\t%.3f" % (t, i_uc))
print("="*10)
print("program finished")
```

```
if __name__ == "__main__":
    main()
```

编辑于 10-28

[文本分类](#) [计算语言学](#) [语料库](#) [文本挖掘](#)

文章被以下专栏收录



自然语言处理技术

对文本分类相关技术的总结

关注专栏

推荐阅读

Python 自然语言处理：轻松上手文本分类！


背景介绍文本分类是NLP中的常见的重要任务之一，它的主要功能就是将输入的文本以及文本的类别训练出一个模型，使之具有一定的泛化能力，能够对新文本进行较好地预测。它的应用很广泛，在很...

程序员 发表于Pytho...

弱监督Snorkel实战NLP文本分类

本文是作者一个tweet/微博文本分类实战项目的全程重现与总结。该项目的最大特点是使用了弱监督技术（Snorkel）来获得海量标注数据，同时使用预训练语言模型进行迁移学习。项目的主要步骤如...

汇智网



基于Fa

戏院小二

还没有评论

写下你的评论...

😊