京东-贪心 NLP 项目实验手册

项目 1：基于京东图书的文本分类

项目设计和编写: 修晔良 姜冰钰

后期编辑: 李文哲

单 位: 贪心科技

2020 年 6 月 22 日

目 录

1 项目描述与目标

文本分类作为自然语言处理领域最经典的技术之一，有着非常广泛的应用，如情感分析、情绪识别、主题分类等等。文本分类任务通常分为两大类，**单标签分类**任务和**多标签分类**任务。单标签分类任务指的是对于一个输入文本，我们需要输出其中的一个类别。举个例子，我们把每一篇新闻分类成一个主题（如体育或者娱乐）。相反，多标签分类任务指的是对于一个输入文本，输出的类别有多个，如对应一篇新闻可以同时输出多个类别：体育、娱乐和音乐。其中，单标签任务又可以分为**二元（binary）分类**和**多类别分类**，二元分类指的是只有两种不同的类别。

在本项目中，我们主要来解决文本单标签的任务。数据源来自于京东电商，任务是基于图书的相关描述和图书的封面图片，自动给一个图书做类目的分类。这种任务也叫作多模态分类。在现实应用中，为了解决一个问题，我们通常会面对多种类型的数据。比如在自动驾驶，为了有效操控车辆的方向，我们可以借助于传感器数据的同时也可以借助于摄像头的数据。

通过本项目的练习，你能通晓机器学习建模的各个流程：

- **文本的清洗和预处理**：这是所有 NLP 项目的前提，或多或少都会用到相关的技术。
- **文本特征提取**：任何建模环节都需要特征提取的过程，你将会学到如何使用 `tfidf`、常用的词向量 [2]、`FastText` [12] 等技术来设计文本特征。
- **图片特征提取**：由于项目是多模态分类，图片数据也是输入信号的一部分。你可以掌握如何通过预训练好的 `CNN` [11] 来提取图片的特征，并且结合文本向量一起使用。
- **模型搭建**：在这里你将会学到如何使用各类经典的机器学习分类模型来搭建算法，其中也会涉及到各种调参等技术。除此之外，处理样本不均衡也是一个非常现实且具有挑战的问题。
- **结果的可视化**：很多模型目前都是黑盒子，很难去理解背后的原因。通

过本项目的练习，你将有机会掌握如何对一个复杂模型的结果做一些可视化分析，试图理解背后分类正确或者分类错误的原因。

- **模型的部署**：工作的最后一般都会涉及到模型的部署，在这里你将会学到如何使用 Flask 等工具来部署模型。

同时，通过本项目

- 1. 熟练掌握分词, 过滤停止词等技术
- 2. 熟练掌握训练、使用 `tfidf`、word2vec、fasttext 模型
- 3. 熟练掌握训练 `Xgboost` 模型, 以及常用评价指标, 并熟练掌握 `Grid Search` 调参方法
- 4. 熟练掌握使用 `Flask` 部署模型
- 5. 了解如何处理不均衡分类问题
- 6. 了解如何获取词性、命名实体识别结果
- 7. 了解如何使用 `Resnet`、`Bert`、`Xlnet` 等预训练模型获取 embedding
- 8. 了解深度学习模型代码架构
- 9. 了解如何使用 `VAE` 获取词嵌入表示
- 10. 了解什么是可解释性模型

作为第一个项目，我们的目标是希望帮助大家能够搭建起一个比较完善的文本分类系统，之后遇到类似的任何问题，都可以有能力去攻克。

2 项目数据描述

在本项目中，我们使用的是京东图书数据。在京东网站上，每一本图书都录属于列表中的一个类目，如图 1所示。由于本项目中需要实现的是单标签多类别的分类模型，我们决定使用**图书的二级类目**作为样本的真实标签，比如“中国文学”，“纪实文学”，“青春校园”等。在给定的数据集中，一共包含 33 个不同类别的标签。

图 1: 京东图书类别[1]

对于标签的识别，我们使用两种类型的数据，分别为图书的文本内容简介（如图 2所示）以及图书的封面图（如图 3所示）。

本项目使用的数据集中，一共包括 206316 个训练样本，58948 个验证样本，29474 个测试样本。

3 项目事宜

本项目是基于图书的文本信息和图片信息来解决文本多分类任务。一般的 AI 项目流程可分为数据预处理、文本特征工程、建模和调参、评估以及部署构成。通过本项目的实操，你将会体会到每个环节的细节如何去落地。

3.1 项目的整个框架

整个项目框架如图 4所示。下面对于图中每个模块做简要的描述，具体的细节请参考本文章后续的内容。

- **特征工程**：这块涉及到文本和图片的特征。对于文本的特征，在本项目

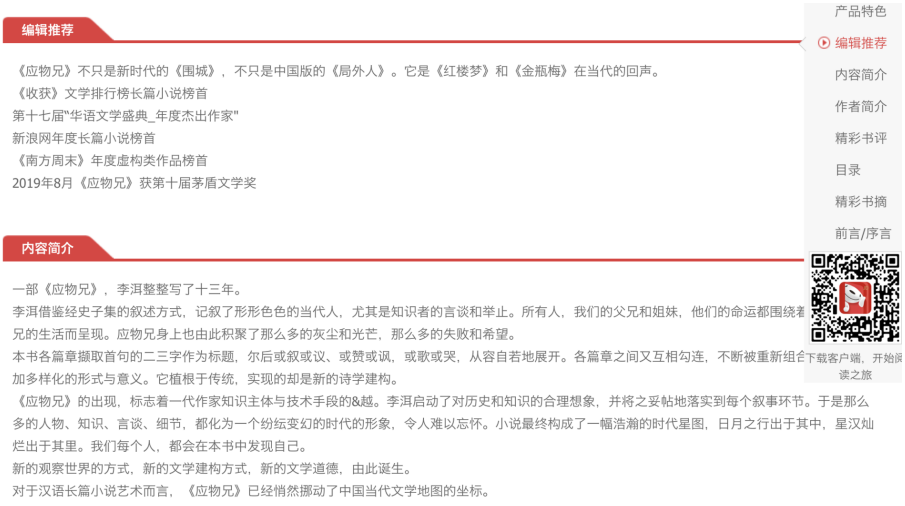


图 2: 京东图书内容简介

图 3: 京东图书封面实例

中需要使用 tf-idf [14], 经典的预训练词向量(FastText, BERT [6])、以及人工抽取的一些特征如单词的词性、实体类别等。对于图片，我们将使用 ResNet [8] 等预训练模型来获得图片向量。等获取完文本和图片向量之后，我们会做特征的拼接来做多模态的训练。

- **模型**：在训练过程中，你将有机会尝试使用各类经典的机器学习模型以及深度学习模型。很多模型已经提供给了大家，大部分模型不需要自己编写。
- **调参**：对于模型的调参环节，我们选择使用网格搜索和贝叶斯优化搜索算法。后者相比前者可以缩小搜索空间，但同时也会增加每次的搜

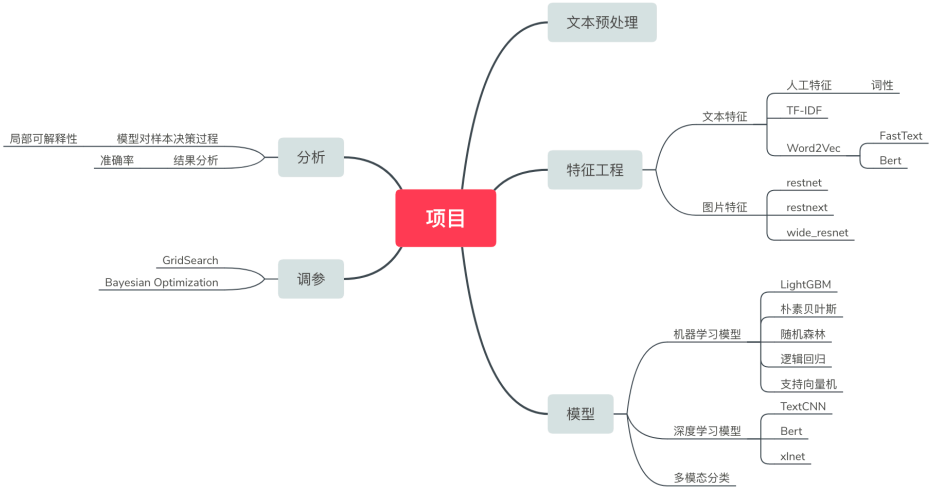


图 4: 京东图书分类项目模块架构图

索代价，具体效率可以通过实验来体会。

- **分析**：评估模型的好坏通常都需要一个标准如准确率或者 F1-Score。除此之外，目前对于深度学习模型来讲，很多时候都以黑盒子的形态来存在，我们很难判断为什么这个模型对于某些样本分类的很准确，对于某些样本分类错误。为了了解更多内部的机理，我们实现了基于词、短语以及词与词、短语与短语之间交互的可解释性模型，即针对一个样本，找出该样本中哪些词或者词组对模型分类的贡献度较大。
- **数据的不平衡处理**：另外，项目数据本身的类别不平衡：有些类别很多，有些类别则比较少，这对于训练模型来讲会有一定的挑战。项目中将使用 BalancedBaggingClassifier、SMOTE [4] 和 ClusterCentroids[3] 等模型来解决上述问题。

4 项目作业描述

任何 AI 项目通常包含数据清洗、特征提取、建模、评估以及部署的环节。所以本项目的安排也遵循此框架。由于具体涉及到的内容偏多，我们将会把上述项目**分成 3 个任务**来给到大家，由浅入深。在第一个任务中，大家需要用到经典的方法先搭建完整的流程，而且能够看到效果，类似于精益创业

中的 MVP（minimum value product）版本。在之后的第二个、第三个任务中，我们会逐步完善项目，最后并部署到线上。

4.1 第一次作业

作为第一个任务，我们希望帮助大家尽快搭建一个完整的流程，能在最短时间内看到效果。所以在第一个任务中，我们所采用的都是经典的方法论。第一次作业的框架图如图5所示。

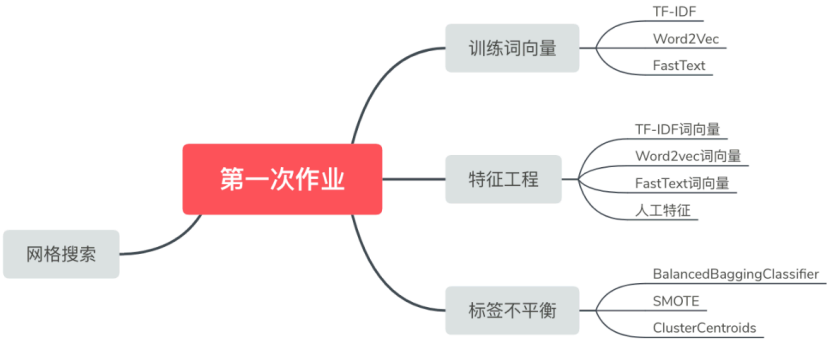


图 5: 第一次作业架构图

4.2 第一次作业描述

对于本次任务，需要完成如下的部分：

- 第一：利用 TF-IDF、Word2Vec [13]、FastText 等模型训练出词向量并训练传统机器学习模型
- 第二：灵活使用文本领域的特征工程方法，为样本提炼出更精确的表示，并分析出这些额外的特征是否让机器学习模型（比如 GDBT [7]）的效果有所提升。
- 第三：处理数据中标签不平衡的问题
- 第四：使用 Grid Search 进行参数搜索优化

4.2.1 第一次作业涉及到的相关技术

训练词向量

需要通过 TF-IDF、Word2vec 和 FastText 方法来获取样本的词嵌入(embedding)。实验中将所有的训练集，验证集和测试集拼接在一起训练，来获取词向量。

```
self.data = pd.concat([
    pd.read_csv(root_path + '/data/train_clean.tsv', sep='\t'),
    pd.read_csv(root_path + '/data/dev_clean.tsv', sep='\t'),
    pd.read_csv(root_path + '/data/test_clean.tsv', sep='\t')
])
```

TF-IDF 的特征可以通过 TfidfVectorizer 对象来训练。

```
count_vect = TfidfVectorizer(
            stop_words=self.stopWords, max_df=0.6, ngram_range=(1, 2))
```

对于 Word2Vec 和 FastText，我们可以通过调用 gensim.models.Word2Vec() 和 gensim.models.FastText() 来训练并获取对应的词向量。

```
self.w2v = models.Word2Vec(min_count=2,
                           window=3,
                           size=300,
                           sample=6e-5,
                           alpha=0.03,
                           min_alpha=0.0007,
                           negative=15,
                           workers=4,
                           iter=10,
                           max_vocab_size=50000)
```

训练完成后，将训练好的词向量进行存储，以待后续使用。

```
logger.info('save tfidf model')
joblib.dump(self.tfidf, root_path +'/src/Embedding/models/tfidf_model')
logger.info('save word2vec model')
self.w2v.save(root_path + '/src/Embedding/models/w2v_model_50000')
```

训练传统机器学习模型

在任务 1 中，我们使用了随机森林模型、支持向量机、朴素贝叶斯、线性回归和 GBDT 模型。这里以 GBDT 模型为例进行一些说明：下面的表格 67分别给出了传统机器学习模型分别在 word2vec 的原始词向量和 FastText 的

原始词向量上的准确率。根据实验结果分析发现，支持向量机模型的准确率最高，随机森林的准确率较低，随机森林模型可以通过调节超参数获得更高的准确率。另外，基于 FastText 的词向量表示的分类模型结果稍低于基于 Word2vec 的试验结果。这是由于 Word2vec 模型训练字典中有 26119 个词，而 FastText 模型中仅有 19592 个词。因此 FastText 构造的样本表示质量要稍低于 Word2vec，从而导致基于 FastText 的分类结果略低于基于 Word2vec 的分类模型。

	#Train.acc	#Test.acc
RandomForestClassifier	0.4661	0.4278
LogisticRegression	0.7171	0.6218
MultinomialNB	0.4459	0.4477
SVC	0.7972	0.7076
LightGBM	0.7477	0.6458

图 6: 机器学习模型在 word2vec 向量中的

	#Train.acc	#Test.acc
RandomForestClassifier	0.4443	0.4156
LogisticRegression	0.706	0.5525
MultinomialNB	0.3859	0.3732
SVC	0.7597	0.6565
LightGBM	0.7317	0.6222

图 7: 机器学习模型基于 Fast 的词向量分类精度

特征工程

对于特征工程，我们做了如下两方面提取的操作：1. **基于词向量的特征工程** 2. **基于人工定义的特征**。

基于词向量的特征工程主要包括以下几个方面：

- 基于 Word2vec 或者 FastText 的词嵌入求出某个词向量的最大值和平均值，并把它们作为样本新的特征。

- 在样本表示中融合 Bert,XLNet [19] 等预训练模型的 embedding。
- 由于之前抽取的特征并没有考虑词与词之间交互对模型的影响，对于分类模型来说，贡献最大的不一定是整个句子，可能是句子中的一部分，如短语、词组等等。在此基础上我们使用大小不同的滑动窗口 (k=[2, 3, 4])，然后进行平均或取最大操作。
- 在样本表示融合样本在自动编码器（AutoEncoder [3]）模型产生的 Latent features。
- 在样本表示融合样本在 LDA [1] 模型产生的 Topic features。
- 将 Word2Vec、Fasttext 词向量求和或取最大值
- 由于没有考虑类别的信息，因此我们从训练好的模型中获取到所有类别的 embedding，与输入的 word embedding 矩阵相乘，对其结果进行 softmax 运算，对 attention score 与输入的 word embedding 相乘的结果求平均或者取最大。具体架构示意图如图 8所示。

示例:

如 input 为： "《应物兄》不只是新时代的《围城》，不只是中国版的《局外人》。", 分词后结果假设为: "《应物兄》不只是新时代的《围城》，不只是中国版的《局外人》。", 过滤掉停止词后结果假设为: " 应物兄只是新时代围城只是中国版局外人", 共计 9 个词。匹配我们已经训练好的 embedding, 得到 9 * 300 维的向量。

因为 input 的句子长短是不一样的, 所以为了保证输入到模型的维度是相同的, 有两种方法: 1. 将长度的维度消去; 2. 将所以文本的的长度补至一样长。第二种方法, 会增加不必要的计算量, 所以在此我们选择使用第一种方法。使用 avg, max 的方法聚合, 得到 300 维的向量。

接下来我们使用类似 n-gram 的方法来获取词组, 短语级别的信息。如我们只考虑前面一个词, 得到结果为: " 应物兄只是只是新新时代时代围城围城只是只是中国中国版版局外人", 8 * 300 或 8 * 2 * 300 维的向量。同样的方法我们将表示长度的维度消去 (由于我们分别考虑前面 2 个词、3 个词、4 个词，所以维度也是相同的, 可以不用消去，而是将 2 * 300 转成 1 * 600 的向量, 与其他特征拼接)。

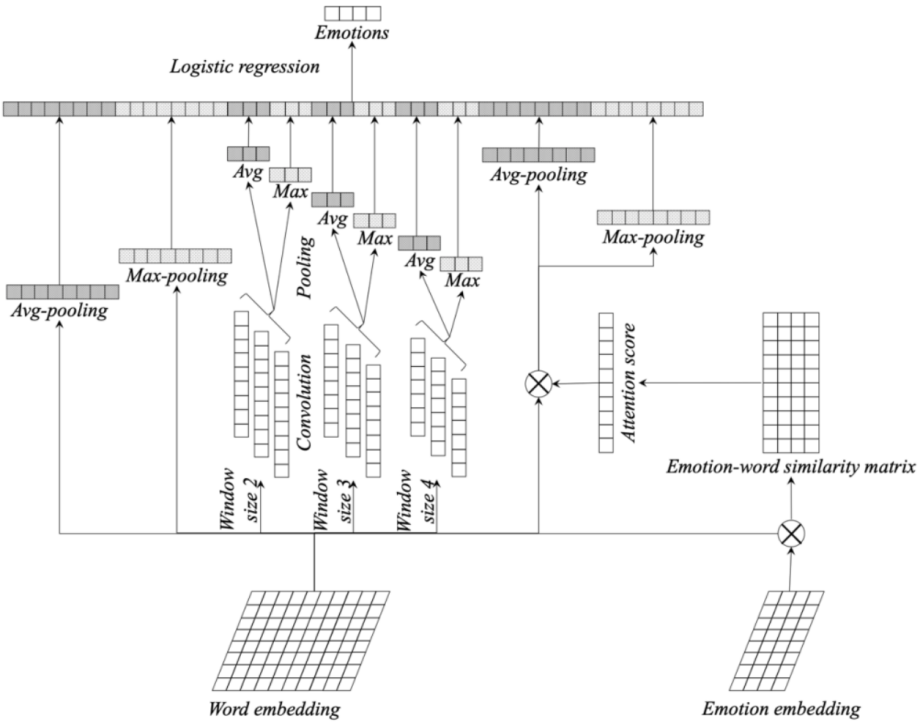


图 8: 特征工程示意图[2]

由于我们的模型没有利用到 label 信息, label 词大多出现在我们的数据集中, 我们考虑使用输入与 label 的相似程度来加权聚合我们的词向量。首先，输入 embedding(假设 9 * 300) 与标签 embedding (假设 30 * 300) 进行矩阵乘法, 得到 (9 * 5) 的矩阵。然后使用 `avg`、`max`、`softmax` 等聚合方法消去标签的维度, 其结果与输入 embedding 进行点乘, 并对得到加权后的结果聚合。将所有特征拼接至一起, 输入至 `Xgboost` 模型训练。

基于人工定义的特征包括以下几个方面:

- 考虑样本中词的词性，比如句子中各种词性 (名词，动词) 的个数，从而使得构造的样本表示具有多样性，从而提高模型的分类精度。
- 通过命名实体识别的技术来识别样本中是否存在地名，是否包含人名等，可以将这些特征加入到样本特征中。

标签不平衡处理

由于本项目数据集中的标签类别较多，但是不同类别样本的数量差异较大，即存在标签不平衡的情况。为了解决该问题，引入了 BalancedBaggingClassifier、SMOTE 和 ClusterCentroids。

方法一：通过上采样的方法从原始数据集中抽样出平衡的数据集。

```
from imblearn.over_sampling import SMOTE
X_resampled, y_resampled = SMOTE().fit_sample(Train_features, Train_label)
```

方法二：通过下采样的方法从原始数据集中抽样出平衡的数据集。

```
from imblearn.under_sampling import ClusterCentroids
cc = ClusterCentroids(random_state=0)
X_resampled, y_resampled = cc.fit_sample(Train_features, Train_label)
```

方法三：通过集成的方法从原始数据集中抽样出样本标签平衡的数据集。

```
from imblearn.ensemble import BalancedBaggingClassifier
print("通过 ensember 中集成的方法来处理不平衡的标签")
bbc = BalancedBaggingClassifier(base_estimator=RF_model,
                                sampling_strategy='auto',
                                replacement=False,
                                random_state=0)
```

网格搜索超参数优化

GBDT 的实现有很多种，在此我们使用微软开发的 LightGBM [9]。GDBT 的超参数较多，为了找到模型最优的超参数组合，我们在项目中使用基于网格搜索的超参数优化算法来实现交叉验证。

1. 网格搜索：网格搜索优化需要提前定义好各个超参数的范围，然后遍历所有超参数组成的笛卡尔积的参数集合。通常网格优化的时间复杂度较大，消耗时间较大。

```
# 网格搜索
parameters = {
    'max_depth': [5, 10, 15, 20, 25],
    'learning_rate': [0.01, 0.02, 0.05, 0.1, 0.15],
    'n_estimators': [100, 500, 1000, 1500, 2000],
    'min_child_weight': [0, 2, 5, 10, 20],
    'max_delta_step': [0, 0.2, 0.6, 1, 2],
```

```
    'subsample': [0.6, 0.7, 0.8, 0.85, 0.95],
'colsample_bytree': [0.5, 0.6, 0.7, 0.8, 0.9],    'reg_alpha': [0, 0.25,
                                                    0.5, 0.75, 1],
    'reg_lambda': [0.2, 0.4, 0.6, 0.8, 1],
    'scale_pos_weight': [0.2, 0.4, 0.6, 0.8, 1]
}
gsearch = GridSearchCV(model, param_grid=parameters, scoring='accuracy',
                                          cv=3)
```

4.3 第二次作业

第二个任务的主要工作是实现多模态模型，也就是加入图片信息的特征。另外，在第二个任务中，你需要使用贝叶斯超参数优化算法来解决优化问题，并且跟经典的网格搜索算法做比较。第二次作业的框架如图 9所示。

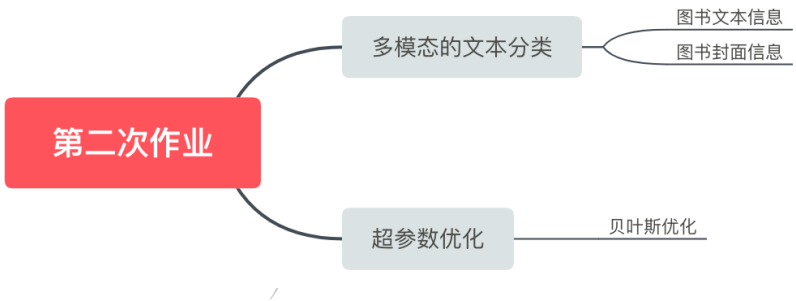


图 9: 第二次作业思维导图

4.3.1 第二次作业描述

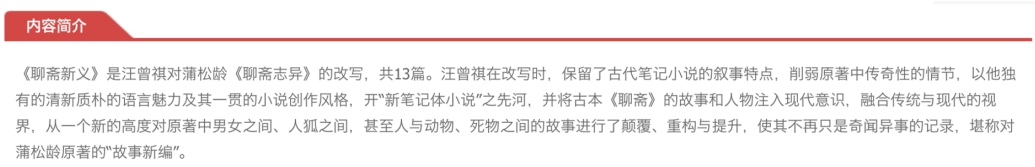
对于本次任务，需要完成如下的思维导图部分。

- **第一**：在第一个任务的基础上，融合图书样本的封面信息来实现多模态的文本多分类。
- **第二**：利用贝叶斯优化（bayesian optimization [16]）来实现对模型超参数的优化。

4.3.2 第二次作业涉及到的相关技术

实现多模态的文本多分类模型

对于图书，我们除了拥有文本信息 10, 还拥有它的封面信息如图 11所示。我们希望把图书的特征信息也融合到模型当中，所以这里的核心问题是如何抽取图片的特征信息。


内容简介

《聊斋新义》是汪曾祺对蒲松龄《聊斋志异》的改写，共13篇。汪曾祺在改写时，保留了古代笔记小说的叙事特点，削弱原著中传奇性的情节，以他独有的清新质朴的语言魅力及其一贯的小说创作风格，开"新笔记体小说"之先河，并将古本《聊斋》的故事和人物注入现代意识，融合传统与现代的视界，从一个新的高度对原著中男女之间、人狐之间，甚至人与动物、死物之间的故事进行了颠覆、重构与提升，使其不再只是奇闻异事的记录，堪称对蒲松龄原著的"故事新编"。

图 10: 图书的文本信息

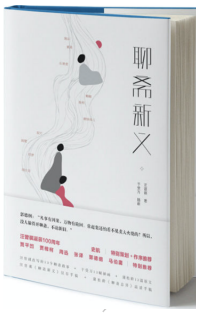


图 11: 图书的封面信息

对于图书的特征提取，我们考虑了各类 Pre-trained 模型，如 restnet、resnext [18]、wide resnet [20] 等 SOTA 模型。模型 [10] 架构如图 12所示。具体的实现细节，请参考实训平台的代码注解。

```
class ImageEncoder(nn.Module):
def __init__(self, args):
    super(ImageEncoder, self).__init__()
    self.args = args
    if self.args.pre-train == 'resnet':
        model = torchvision.models.resnet152(pretrained=True)
    elif self.args.pre-train == 'resnext101':
        model = torchvision.models.resnext101_32x8d(pretrained=True)
    elif self.args.pre-train == 'wide_resnet':
        model = torchvision.models.wide_resnet101_2(pretrained=True)
    modules = list(model.children())[:-2]
    self.model = nn.Sequential(*modules)
```

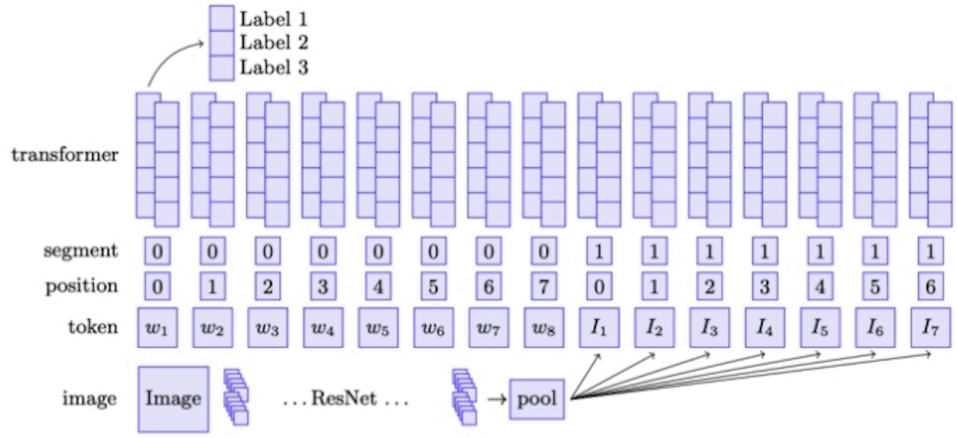


图 12: 图书封面特征信息的提取 [17]

```
pool_func = (             nn.AdaptiveAvgPool2d
             if args.img_embed_pool_type == "avg"
             else nn.AdaptiveMaxPool2d
         )

         if args.num_image_embeds in [1, 2, 3, 5, 7]:
             self.pool = pool_func((args.num_image_embeds, 1))
         elif args.num_image_embeds == 4:
             self.pool = pool_func((2, 2))
         elif args.num_image_embeds == 6:
             self.pool = pool_func((3, 2))
         elif args.num_image_embeds == 8:
             self.pool = pool_func((4, 2))
         elif args.num_image_embeds == 9:
             self.pool = pool_func((3, 3))

     def forward(self, x):
         # Bx3x224x224 -> Bx2048x7x7 -> Bx2048xN -> BxNx2048
         out = self.pool(self.model(x))
         out = torch.flatten(out, start_dim=2)
         out = out.transpose(1, 2).contiguous()
         return out
```

贝叶斯优化基于贝叶斯优化的调参算法由 J.Snoek(2012) 提出。它的主要思想是，给定优化的目标函数 (广义的函数，只需指定输入和输出即可，无需

知道内部结构以及数学性质)，通过不断地添加样本点来更新目标函数的后验分布 (高斯过程, 直到后验分布基本贴合于真实分布。简单的说，就是考虑了上一次参数信息，从而更好的调整当前的参数。

```
def Bayes_Optimization(Train_features4, Test_features4,

                                    Train_label4, Test_label4):
    rf_bo = BayesianOptimization(
        rf,
        {'n_estimators': (10, 2000),'max_depth': (4, 15),
         'reg_alpha': (0, 50),'reg_lambda': (0, 100),
         'learning_rate': (0.01, 0.1),
         'subsample': (0.1, 0.99)
         }
    )
    rf_bo.maximize()
```

贝叶斯优化与常规的网格搜索或者随机搜索的区别是：贝叶斯调参采用高斯过程，考虑之前的参数信息，不断地更新先验；网格搜索未考虑之前的参数信息。贝叶斯调参迭代次数少，速度快；网格搜索速度慢，参数多时易导致维度爆炸；贝叶斯调参针对非凸问题依然稳健；

4.4 第三次作业

第三次作业的核心目标是训练深度学习的分类模型，同时通过一种方式来解释深度学习模型。很多时候，我们把深度学习当作黑盒子来对待，其实不太清楚里面的运行机理。如一个样本被分类错误了，我们其实也不清楚为什么。为了解开模型内部的黑纱，需要一种方法来对内部做一些可解释性的操作。第三次作业框架如图 13所示。

4.4.1 第三次作业描述

对于本次任务，需要完成如下的部分：

- **第一**：在第一次和第二次作业的基础上，实现文本分类的深度学习模型（比如 TextCNN）。

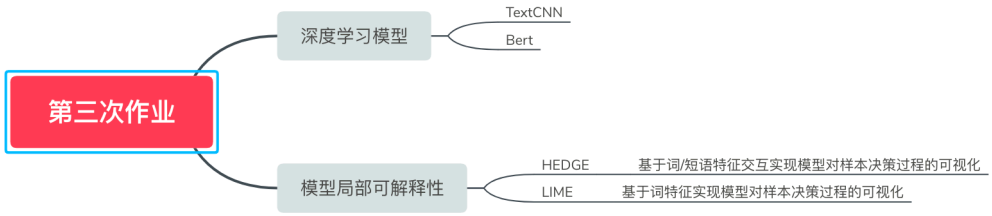


图 13: 第三次作业架构图

- **第二**：实现模型对样本的决策过程的可视化。

4.4.2 第三次作业涉及到的相关技术

训练深度学习模型

在本项目中，我们实现了一些经典的深度学习方法，其中包括 CNN、Bert 等。利用 Pre-trained Bert 模型，在图书案例上进行 fine-tune。对比实验结果可以发现，基于神经网络模型的分类效果一般要高于传统的机器学习模型的效果。

```
class Model(nn.Module):
    def __init__(self, config):
        super(Model, self).__init__()
        model_config = BertConfig.from_pretrained(
            config.bert_path, num_labels=config.num_classes)
        self.bert = BertModel.from_pretrained(
            config.bert_path, config=model_config)
        for param in self.bert.parameters():
            param.requires_grad = True
        self.fc = nn.Linear(config.hidden_size, config.num_classes)


    def forward(self, x):
        context = x[0]  # 输入的句子
        # 对 padding 部分进行 mask，和句子一个 size，padding 部分用 0
                                                表示，如：[1, 1, 1, 1, 0, 0]
        mask = x[1]
```

```
        token_type_ids = x[2]
_, pooled = self.bert(context,
                                                attention_mask=mask,
                                                token_type_ids=token_type_ids)
        out = self.fc(pooled)
        return o
```

实现模型对样本决策的可解释性

神经网络模型通常被看成一个黑盒，比如在文本分类任务上，即很难直观上理解为什么样本会被模型预测为对应的类别。为了解决该问题，可以通过局部可解释性的方法来完成模型对样本预测结果的解释和可视化。局部可解释性主要是通过分析改变样本的特征从而导致样本预测结果变化的程度，如果该特征对样本决策影响较大的话，则说明该特征对样本分类起到关键作用，反之，如果该特征对样本预测结果的影响不大，则说明该特征对样本决策的贡献度较大。

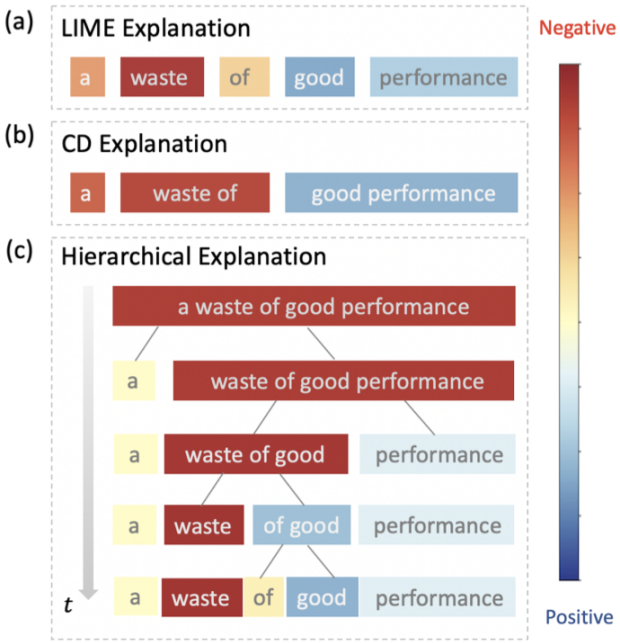


图 14: 文本分类中的局部可解释性 [5]

如图 14所示，(a) 显示的方法主要是分析在词级别上模型对样本决策的可解

释性(LIME [15]),(b)给出了样本在分类模型下短语级别的可解释性,(c)给出了样本在分类模型下不同词或短语交互情况下的可解释性 (HEDGE [5])。以（a）为例进行说明，词的不同颜色体现了该词对样本在模型下分类预测成对应类的贡献度。

参考文献

[1] Blei, D. M., Ng, A. Y., and Jordan, M. I. Latent dirichlet allocation. *Journal of machine Learning research 3*, Jan (2003), 993–1022.

[2] Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics 5* (2017), 135–146.

[3] Bourlard, H., and Kamp, Y. Auto-association by multilayer perceptrons and singular value decomposition. *Biological cybernetics 59*, 4-5 (1988), 291–294.

[4] Bowyer, K. W., Chawla, N. V., Hall, L. O., and Kegelmeyer, W. P. SMOTE: synthetic minority over-sampling technique. *CoRR abs/1106.1813* (2011).

[5] Chen, H., Zheng, G., and Ji, Y. Generating hierarchical explanations on text classification via feature interaction detection. *arXiv preprint arXiv:2004.02015* (2020).

[6] Devlin, J., Chang, M., Lee, K., and Toutanova, K. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR abs/1810.04805* (2018).

[7] Friedman, J. H. Greedy function approximation: a gradient boosting machine. *Annals of statistics* (2001), 1189–1232.

[8] He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *CoRR abs/1512.03385* (2015).

[9] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in neural information processing systems* (2017), pp. 3146–3154.

[10] Kiela, D., Bhooshan, S., Firooz, H., and Testuggine, D. Supervised multimodal bitransformers for classifying images and text. *arXiv preprint arXiv:1909.02950* (2019).

[11] Kim, Y. Convolutional neural networks for sentence classification. *CoRR abs/1408.5882* (2014).

[12] Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[13] Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. Distributed representations of words and phrases and their compositionality. *CoRR abs/1310.4546* (2013).

[14] Paik, J. H. A novel tf-idf weighting scheme for effective ranking. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval* (2013), pp. 343–352.

[15] Ribeiro, M. T., Singh, S., and Guestrin, C. "why should I trust you?": Explaining the predictions of any classifier. *CoRR abs/1602.04938* (2016).

[16] Snoek, J., Larochelle, H., and Adams, R. P. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems* (2012), pp. 2951–2959.

[17] Tsai, Y.-H. H., Bai, S., Liang, P. P., Kolter, J. Z., Morency, L.-P., and Salakhutdinov, R. Multimodal transformer for unaligned multimodal language sequences. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (Volume*

1: Long Papers) (Florence, Italy, 7 2019), Association for Computational Linguistics.

[18] Xie, S., Girshick, R. B., Dollár, P., Tu, Z., and He, K. Aggregated residual transformations for deep neural networks. *CoRR abs/1611.05431* (2016).

[19] Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., and Le, Q. V. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems* (2019), pp. 5754–5764.

[20] Zagoruyko, S., and Komodakis, N. Wide residual networks. *arXiv preprint arXiv:1605.07146* (2016).