Assignment 1-2

Files

- data/
  - 数据存放目录
- model/
  - 模型存放目录
- logs/
  - 日志存放目录
- app.py

    - 代码部署部分
- src/
    - 核心代码部分
- `src/data`
    - `src/data/dataset.py`：主要用于深度学习的数据处理
    - `src/data/mlData.py`：主要用于机器学习的数据处理
    - `src/data/dictionary.py`：生成词表， 能够根据id确定词， 或者根据词确定id
- `src/word2vec/`
    - `src/word2vec/embedding.py`: tfidf, word2vec, fasttext, lda 的训练，保存加载。
    - `src/word2vec/autoencoder.py`: autoencoder的训练，保存加载。
- `src/utils/`
    - `src/utils/config.py`: 相关参数的配置文件， 如训练数据集所在目录， DL模型相关参数等等
    - `src/utils/feature.py`: 特征工程相关的函数
    - `src/utils/tools.py`: 通用类函数
- `src/ML/`
    - `src/ML/fasttext.py`: fasttext 模型训练，预测， 保存
    - `src/ML/main.py`: 机器学习类模型总入口
    - `src/ML/model.py`: 包含特征工程，参数搜索， 不平衡处理， lightgbm的预测
- `src/DL/`
    - `src/DL/train.py`: 深度学习模型训练主入口
    - `src/DL/train_helper.py`: 深度学习模型实际训练函数
- `src/DL/models`:
    - 深度学习模型

TO-DO list:

必备资料

为了完成以下任务，我们需要逐步熟悉、掌握pandas, sklearn, lightgbm， gensim等工具，所以请大家在完成每一个模块时先查阅一下pandas的API文档,sklearn的API文档,gensim的API文档,lightgbm的API文档，弄清楚要使用的模块是做什么的以及如何使用。

模块1: 训练Embedding

src/word2vec/embedding.py

任务1: 完成tfidf训练。

使用sklearn 训练tfidf 模型， 注意了解模型各参数的意义。

任务2: 完成word2vec训练。

使用gensim 训练 word2vec 模型， 注意了解API 各参数的意义。

任务3: 完成fasttext训练。

使用gensim 训练 fasttext 模型， 注意了解API 各参数的意义。

任务4: 完成lda训练。

使用gensim 训练 lda 模型， 注意了解API 各参数的意义。

任务5: 完成autoencoder训练。

根据 `src/word2vec/autoencoder.py`的实现， 训练 lda 模型， 注意类中间的调用。

测试

```
cd src/word2vec
python3 embedding.py
```

你可能用到的模块:

```
sklearn.feature_extraction.text.TfidfVectorizer

gensim.models
```

模块2: 预处理阶段

src/data/mlData.py:

任务1: 处理文本数据。

1. 对文本进行分词
2. 去除停止词

3. 将文本label转换为id

src/ML/models.py:

任务2: 加载模型。

1. 使用torchvision 初始化 resnet152模型
2. 使用torchvision 初始化 resnext101_32x8d 模型
3. 使用torchvision 初始化 wide_resnet101_2 模型
4. 加载bert 模型

你可能用到的模块:

```
jieba.cut
torchvision.models.resnet152
torchvision.models.resnext101_32x8d
torchvision.models.wide_resnet101_2
```

模块3: 特征工程

src/ML/models.py & src/utils/feature.py

任务1: 获取 基本的 NLP feature

1. 完成以下basic NLP 特征

```
def get_basic_feature(df):
    计算input 长度
    计算大写 个数
    计算大写个数和长度的比值
    计算感叹号的个数
    计算问号长度
    计算标点符号个数
    计算 `*&$%` 符号的个数
    计算词的个数
    计算唯一词的个数
    词的个数与唯一词个数的比例
    计算名词个数与长度的占比
```

```
    计算形容词个数与长度的占比
    计算动词个数与长度的占比
    计算名词个数与词的个数的占比
    计算形容词个数与词的个数的占比
    计算动词个数与词的个数的占比
    计算title的词的个数
    计算词的平均长度
    计算标点个数与词的个数的占比
    return df
```

2. 在 `src/ML/models.py` 中的 `feature_engineer`函数 调用 `src/utils/feature.py`中的 `get_basic_feature`函数

任务2: 根据已经加载的模型获取封面的特征

1. 完成 image 特征的获取

```
def get_img_embedding(cover, model):
    transforms = get_transforms()
    1. 读取封面， 返回modal embedding
    hint  返回需要转换成cpu模式
    return
```

2. 在 `src/ML/models.py` 中的 `feature_engineer`函数 调用 `src/utils/feature.py`中的 `get_img_embedding`函数

任务3: 根据已经加载的bert模型获取embedding

1. 完成bert embedding 特征的获取

```
def get_pretrain_embedding(text, tokenizer, model):
    1. 返回bert embedding
    hint  返回需要转换成cpu模式
    return
```

2. 在 `src/ML/models.py` 中的 `feature_engineer`函数 调用 `src/utils/feature.py`中的 `get_pretrain_embedding`函数

任务4: 根据`src/data/mlData.py` 中已经加载的`lda` 模型， 获取lda 特征

1. 将输入转换为bag of word 格式
2. 完成lda 特征的获取

```
def get_lda_features(lda_model, document):
    1. 返回lda feature
    return
```

3. 在 `src/ML/models.py` 中的 `feature_engineer`函数 调用 `src/utils/feature.py`中的 `get_lda_features`函数

任务4: 根据`src/data/mlData.py` 中已经加载的 `autoencoder` 模型， 获取 autoencoder 特征

1. 完成 autoencoder 特征的获取

```
def get_autoencoder_feature(train,
                        test,
                        max_features,
                        max_len,
                        model,
                        tokenizer=None):
    1. 返回autoencoder embedding
    return
```

2. 在 `src/ML/models.py` 中的 `feature_engineer`函数 调用 `src/utils/feature.py`中的 `get_autoencoder_feature`函数

你可能用到的模块:

```
transformers.BertModel
transformers.BertTokenizer
lightgbm
```

模块4: 模型训练

src/ML/models.py:

任务1: 不平衡数据处理。unbalance_helper 函数

1. 定义 over_sampling 方法， 如SMOTE， 处理样本不平衡问题
2. 定义 under_sampling 方法， 如ClusterCentroids， 处理样本不平衡问题

任务2: 参数搜索。

1. 使用 param_search 进行参数搜索
2. 将搜索到的参数， 使用`set_params` 进行更新

任务3: 模型评价。

1. 预测测试集的label
2. 预测训练机的label
3. 计算percision , accuracy, recall, fi_score

任务4: 模型保存以及加载

1. 保存模型 save函数
2. 加载模型 load函数

模块5: 模型部署

src/ML/models.py:

任务1: 完成模型预测函数

1. 将输入数据转化为特征
2. 使用训练好的模型进行预测

app.py

任务2: 使用 `flask`进行部署

1. 接受RESTFul 传递的参数， 调用`src/ML/models.py` 中的`predict` 函数进行预测
2. 使用`curl` 命令或`postman`等工具进行测试

启动

```
python3 app.py
```

你可能用到的模块：

```
Flask
bayes_opt.BayesianOptimization
sklearn.metrics
sklearn.model_selection.GridSearchCV
imblearn.ensemble.BalancedBaggingClassifier
imblearn.over_sampling.SMOTE
imblearn.under_sampling.ClusterCentroids
```

模块6: fine-tune bert 模型 （选做）

src/DL/train.py:

任务1: 加载数据

1. 调用自定义的MyDataset， 并创建DataLoader

src/DL/train.py:

任务2: 完成深度学习的核心代码

1. 初始化AdamW 优化器
2. 加载模型进行训练
3. 清空梯度
4. 计算loss
5. loss backpropergation
6. 优化器step

src/DL/models/bert.py

任务3: 完成`bert`模型的训练

```
    ```
    def forward(self, x):
            ### TODO
            # 构建bert 分类模型
            return out
    ```
```

1. 完成forward 的功能

运行

```
cd src/DL/
python3 train.py --model bert
```