京东-贪心 NLP 项目实验手册

项目 1：基于京东健康-智能分诊的文本生成

项目设计和编写: 姜冰钰

后期编辑: 林培鑫 李文哲

单 位: 贪心科技

2021 年 3 月 5 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。数据源来自于京东健康，任务是基于患者的病情描述，自动给一个门诊科室的分类。

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` [6] 等技术来设计文本特征。
- **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。
- **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

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。
在本项目中，我们使用的是京东健康的分诊数据。互联网医生服务可以构建医生与患者之间的桥梁，京东通过智能分诊项目，可以根据用户提供的文字型的病情描述精准识别，并自动帮助用户判断需要去哪个分诊科室，有效减少在线问诊被反复多次转接的情况发生，提高科室分配的准确度，实现降本增效。

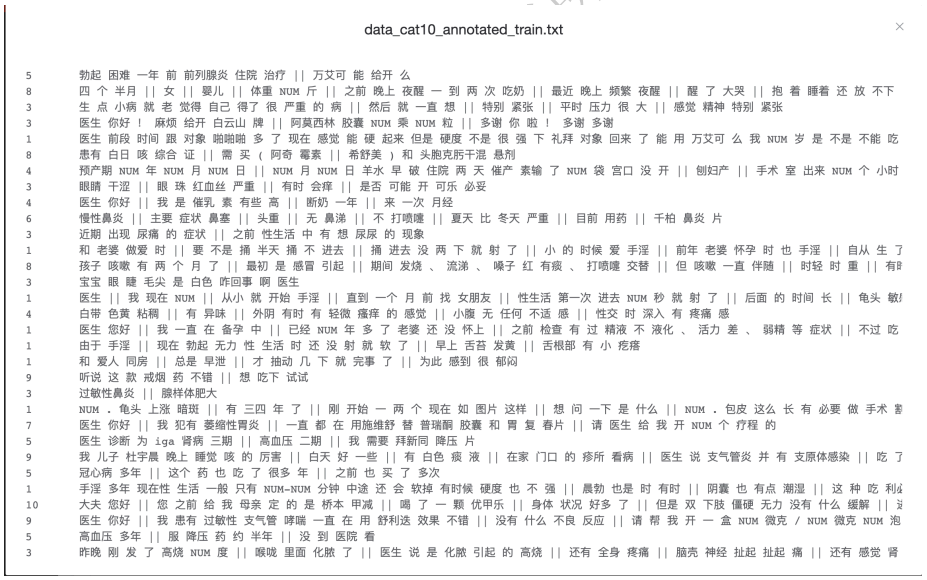

图 1: 智能分诊样本数据

本项目我们主要使用 28000 多条样本数据来训练文本分类模型，在 NeuFoundry 的“京东健康-智能分诊”下可找到，这一数据集下有更多详细描述以及补充数据，可以具体查看。

2 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。

2.1 项目的整个框架

整个项目框架如图 2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。

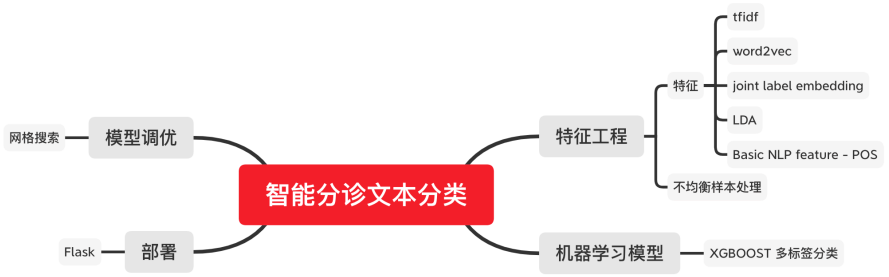

图 2: 京东文本分类项目模块架构图

- **特征工程**：对于文本的特征，在本项目中需要使用 `tf-idf` [7]，经典的预训练词向量（`FastText`, `BERT` [4]）、以及人工抽取的一些特征如单词的词性、实体类别等。
- **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`。

2.2 本项目涉及到的相关技术

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')
```

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 [8] 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 3所示。

示例:

如 input 为: " 以前经常吃多了胃部会不舒服", 分词后结果假设为: ” 以前经常吃多了胃部会不舒服”, 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: ” 以前经常经常吃吃多多了了胃部胃部会会不不舒服”, 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

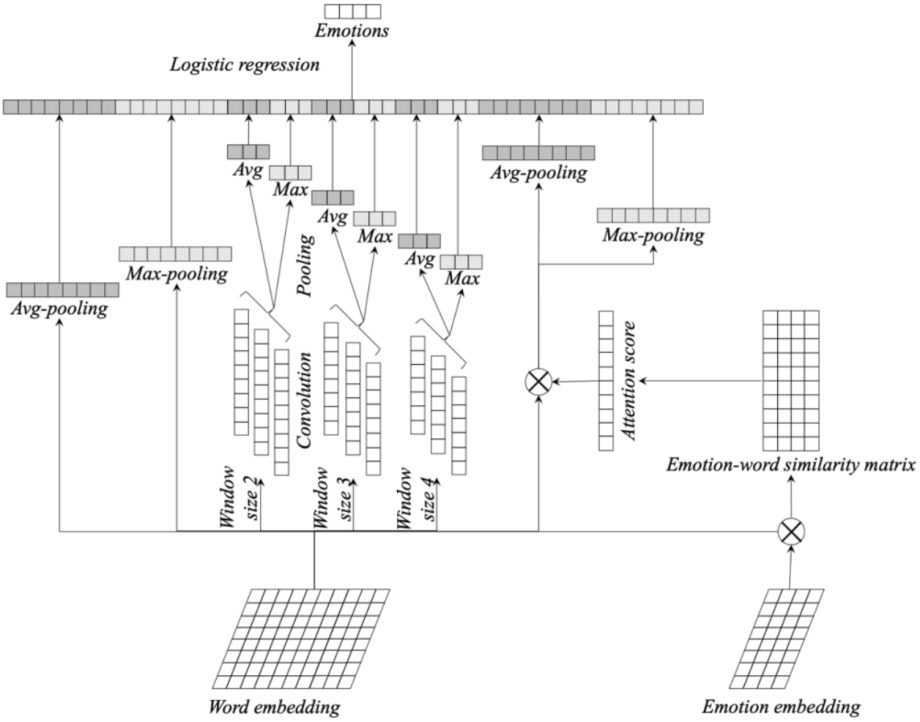

图 3: 特征工程示意图[1]

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)。

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。

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 [5]。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)
```

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

[5] 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.

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

[7] 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.

[8] 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.