小样本数据分类任务

痛！太痛了！

李一鸣
计算机科学与技术&研一
中国科学院计算技术研究所
中国-北京
liyiming22s1@ict.ac.cn

张兆
计算机科学与技术&研一
中国科学院计算技术研究所
中国-北京
zhaozhao809@163.com

李想
模式识别与智能系统&研一
中国科学院自动化研究所
中国-北京
2300049883@qq.com

赵家乐
计算机科学与技术&研一
中国科学院计算技术研究所
中国-北京
1007613549@qq.com

团队简介

“痛！太痛了”团队由 4 名中国科学院大学 2022 级研究生新生组成，其中 3 人来自中国科学院计算技术研究所，1 人来自中国科学院自动化研究所。团队的研究兴趣涉及半监督学习、智能问答、生物信息学等领域。

摘要

近年来，随着政策扶植、国家工业化水平和国民教育水平提高，我国的专利申请量爆发式增长，对于专利文本的分类管理需求也与日俱增。得益于自然语言处理技术的发展，基于 Pretraining-Finetune 的预训练范式在学术界与工业界得到了广泛的应用。在小样本专利文本分类问题中，我们利用百度开源的 ERNIE 3.0 模型，设计了一个基于动态阈值伪标签生成与尾部类别数据增强的小样本文本分类方法，能够较好地适应小样本学习场景。实验结果表明，在 CCF BDCI 小样本数据分类任务上，我们的方法最终在 B 榜测试集达到了 0.5955 的 F1 分数。

关键词

专利文本分类，小样本学习，动态阈值伪标签

引言

近年来，随着科技的进步与创新能力的提升，我国的专利申请量快速增长，专利检索、查新、管理等需求也随之增加。为了满足多元化专利检索需求，提升专利服务质量，通常需要建立多个维度的专利分类体系。常见的分类体系有国际专利分类 (IPC)、联合专利分类 (CPC)、欧洲专利分类 (ECLA) 等，但是这些分类体系比较复杂，专业性强，对非 IP 人员而言使用有一定的困难。近年来，随着大规模预训练模型[1]的发展，自然语言处理技术在工业界逐渐落地，这也为解决专利分类问题带来了新的思路。

相较于传统文本，专利文本（尤其是标注文本）的数据量较为稀少。因此，小样本学习在专利文本分类中至关重要。此外，由于不同领域的专利数量大不相同，某些类别的样本数量明显少于其他类别，数据集标签分布存在长尾现象，传统的小样本学习则较难适应某些长尾类别。

为了解决上述问题，我们提出了**基于类别动态阈值伪标签与尾部类别数据增强的小样本文本分类方法**。在我们的方案中，我们同时考虑到了小样本学习及数据集长尾分布等问题。具体地，与传统伪标签技术预先选定一个固定的阈值 c 不同[2]，我们为每一个类别 i 设置一个单独的阈值 c_i。同时，为了缓解数据不平衡的问题，我们为数据集中的尾部类别应用了两类不同的数据增强方法，以扩充其样本量。实验结果表明，我们的方法在小样本专利文本分类的任务中取得了较好的效果，并最终在初赛 A 榜测试集上达到 0.6516 的 F1 分数，在 B 榜测试集上达到了 0.5955 的 F1 分数。

1 任务描述

本赛题要求对专利文本数据进行分类，训练集包括 958 条专利数据，每一条样本包括专利权人、专利标题、专利摘要、分类标签等属性，其中标签经过脱敏处理，共 36 类。

赛题采用 Macro-F1 指标对于模型性能进行评估，具体定义如公式 (1 − 1) 所示：

$$\langle F1\rangle = \frac{1}{n}\sum_{i}^{n} F1_i = \frac{1}{n}\sum_{i}^{n}\frac{2\cdot P_i\cdot R_i}{P_i + R_i} \qquad (1-1)$$

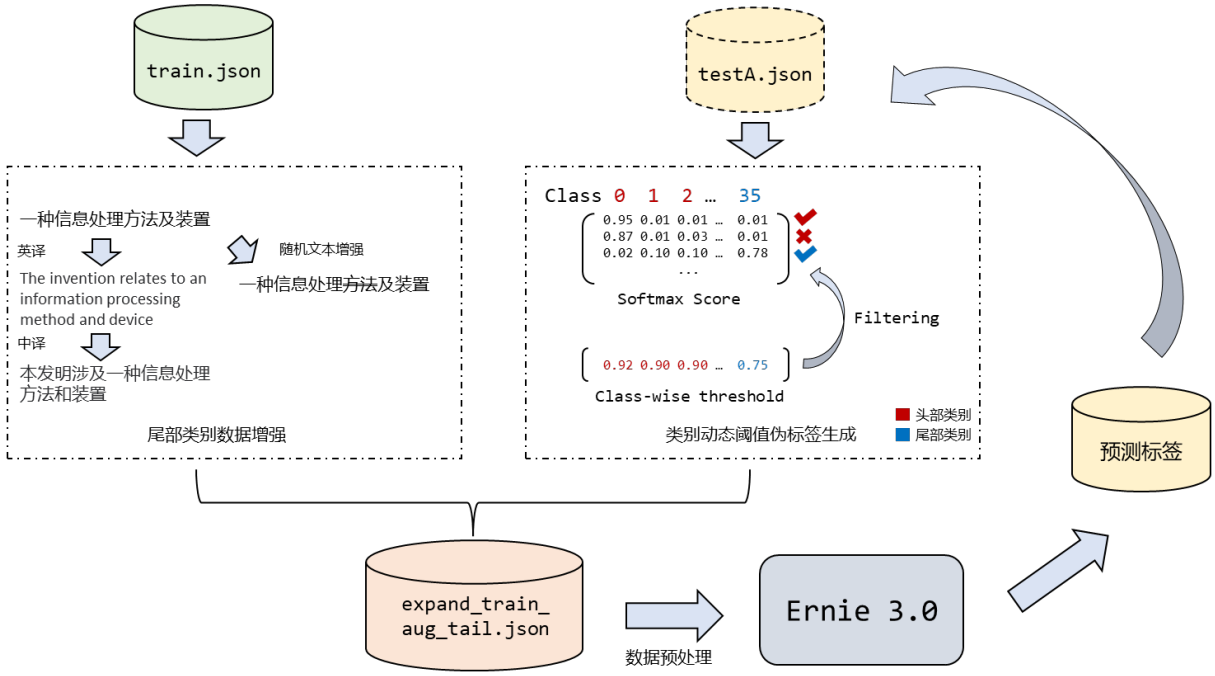

图 1 解决方案整体流程图

其中，P_i 是第 i 个类别的 $Precision$，R_i 是第 i 个类别的 $Recall$，n 是类别总数。对于某一个类别 i 来说，$Precision$ 和 $Recall$ 的计算方式如公式 $(1-2)$ 和 $(1-3)$ 所示：

$$P_i = \frac{TP_i}{TP_i + FP_i} \tag{1-2}$$

$$R_i = \frac{TP_i}{TP_i + FN_i} \tag{1-3}$$

其中，TP_i 为类别 i 的真正例，FP_i 为类别 i 的假正例，FN_i 为类别 i 的假负例。

2 解决方案

图 1 展示了我们解决方案的流程图。具体地，我们首先根据模型在 A 榜测试集上输出的 softmax 分数为不同类别的样本设置不同阈值，来生成伪标签扩充训练集规模，并针对尾部样本进行数据增强，缓解了尾部类别的数据稀疏问题。接着，使用扩充后的训练集对 ernie-3.0-base 语言模型[3]在文本分类任务上进行微调。此外，为了提升模型的鲁棒性以及解决类别不平衡问题，我们进一步采用了对抗训练，提升模型的鲁棒性，实现了更加精准的分类。

2.1 数据预处理

训练及测试数据集的每个样本包含 title、assignee、abstract 字段，这些信息包含不同粒度的信息，为了更好地将各字段组织起来，用于文本分类，我们受近年在自然语言处理领域中新兴的提示学习启发，将各字段重新组织为一段文本，基于以下模板构造模型的输入。针对一条训练样本 json 条目 x，我们将其组织为如下形式："这份专利的标题为《{x.title}》，由{x.assignee}公司申请，详细说明如下：{x.abstract}"

2.2 对抗训练

对抗训练旨在提升模型的鲁棒性，同时也能提高模型的泛化能力，其基本思想是对原始输入样本 x 添加扰动 r_{adv}，得到对抗样本 x_{adv}，再将对抗样本输入模型训练。

具体地，我们采用 FGM (Fast Gradient Method) 攻击算法[4]，其所施加的扰动与样本的梯度方向相同，如公式 $(2-1)$ 和 $(2-2)$ 所示：

$$r_{adv} = \frac{\epsilon g}{||g||_2} \quad (2-1)$$

$$g = \nabla_x L(\theta, x, y) \quad (2-2)$$

生成的对抗样本如公式 $(2-3)$ 所示：

$$x_{adv} = x + r_{adv} \quad (2-3)$$

2.3 伪标签

研究指出在长尾识别任务下，尾部类别虽然召回率 Recall 较低，但却有着较高的精确度 Precision，而头部类别则相反[5]。这启发我们根据标签分布情况及模型预测情况为每一个类别设置不同的阈值：头部类别较多，模型学习情况较好，应为其设置较高的阈值，保证其精确度；尾部类别较少，模型学习情况较差，应为其设置较低的阈值，从而获得较高的召回率。因此，我们为本任务设计了一种类别相关的动态阈值伪标签产生策略，流程如下，对于每一类别，我们先得到预测标签为该类的所有记录（见表 1 算法 1 第三行），之后将这些记录按照对于该类预测概率由大到小进行排序（见表 1 算法 1 第四行），进一步地，我们求其 α 分位数（即前 α %大的预测概率）作为该类别的阈值（见表 1 算法 1 第五行），同时，若求得的阈值小于一个预先给定的固定阈值 γ，则将其重新置为 γ（见表 1 算法 1 第六行）。利用 α 分位数筛选伪标签可以充分地考虑到各个类别的学习情况（头部类别学习情况较好，通常模型预测的置信度较高，而尾部类别则相反），同时固定阈值 γ 作为下限，使得尾部类别的筛选不至于混入过多的噪声样本。

表 1 伪标签生成流程算法

算法 1 伪标签生成流程

输入：M: Softmax 分数矩阵，维度为(N, C)
输入：α: 用于阈值筛选的分位数
输入：γ: 用于阈值筛选的固定阈值
输出：L: 算法得到的各个类别阈值分数

```
L = []
for c = 0 → C − 1 do
    P = M[𝟙[argmaxM == c]][c]
    Sort P in descending order
    L_c = Percentile(P, α)
    if L_c < γ then L_c = γ
    L.append(L_c)
end for
```

值得一提的是，上述的伪标签产生流程可以迭代进行，即训练得到一个较好的模型 A，利用其产生的伪标签再训练一个模型 B，再利用模型 B 对无标签测试数据打伪标签，并由此再训练一个新的模型 C。在实验中，我们分别采用线上 F1 分数为 0.600, 0.599, 0.632, 0.642, 0.646 的（集成）模型为 A 榜测试数据标记伪标签，并将其并入初始训练集中，分别得到伪标签扩展数据集 train_0，train_1，train_2，train_3，train_4，并在最终参赛系统的训练中与尾部数据增强策略配合使用。

2.4 数据增强

为了缓解类别不平衡的问题，我们对尾部类别（类别 12、22、32、35）应用两类数据增强方法。

1. 回译：我们利用谷歌翻译接口，将中文翻译为英法德日韩 5 种语言，再翻译回中文，得到新的样本。

2. 简单文本数据增强[6]：由 4 种不同的方法组成，分别是同义词替换（从句子中随机选择 n 个非停用词，随机选择它们的同义词进行替换）、随机插入（从句子中随机选择一个非停用词的单词，随机选择它的一个近义词并将它插入在句子的任意位置。并将此过程重复 n 次）、随机替换（随机选择句子中的两个单次并将它们交换位置。将此过程重复 n 次）与随机删除（对句子中的每一个单词，都以一给定概率 p 判定此单词是否被删除）。

3 实验结果

3.1 参赛系统

通过前期多种模型的线上 F1 实验，我们最终选择了效果较好的 9 个模型作为参赛系统的最终结果。9 个模型的具体参数细节各不相同，训练数据也采用了不同阶段得到的伪标签，且大多是在经过多折交叉验证后选取的效果较好的某一折或某几折。同时为了满足模型的总大小不超过 2G 的要求，我们将单精度模型转换为半精度，实验表明这种转换对于最终的推理结果影响不大。最后我们通过投票法将 9 个模型集成在一起得到最终的推理结果。9 个模型的具体参数细节如表 2 所示，其中 4 号和 6 号模型选取了其中的两折结果。最终，集成模型在初赛 A 榜测试集上达到 0.6516 的 F1 分数，在 B 榜测试集上达到了 0.5955 的 F1 分数。

表 2 参赛系统模型设置及性能

编号	线上 F1	训练数据	其他参数
1	0.6288	train_0	fgm=True, seed=42
2	0.6308	train_0	fgm=False, seed=909
3	0.6260	train_1	fgm=True, seed=42
4	0.6323	train_2	fgm=True, seed=42
5	0.6329	train_3	fgm=True, seed=42
6	0.6267	train_3	fgm=True, seed=42
7	0.6334	train_4	fgm=True, seed=42

*注：表格编号 5、6 对应实验结果为同一实验配置下 K 折交叉验证的不同折次的结果

3.2 消融实验

3.2.1 预训练模型

目前 NLP 研究界涌现出大量的预训练模型，代表性的开源中文预训练模型也有很多种。我们对搜集到的多种预训练模型进行了多次对比实验，最终的实验结果表明 ERNIE 3.0 更适用于我们的小样本分类学习任务。其他模型的实验结果如表 3 所示，实验均在 NVIDIA GeForce RTX 2080 Ti * 2 上进行，batch_size 设置为 12，模型迭代训练 40 轮。

表 3 不同预训练模型之间的对比

预训练模型	线上 F1
mengzi-bert-base	0.5895
ernie-3.0-base-zh	**0.6120**
chinese-macbert-base	0.5857
chinese-roberta-wwm-ext	0.5955

3.2.2 伪标签技术

在本小节，我们通过对表 1 涉及到的两个超参数 α 及 γ 进行消融实验以证明我们提出的类别相关的动态阈值伪标签产生策略相较于传统的固定阈值伪标签生成更有优势，相关实验结果如表 4 所示，其中，实验编号 1 即为算法 1 的默认参数设置，实验编号 2 的设置即为传统伪标签产生算法（如 FixMatch）等采取的固定阈值策略，实验编号 3 即不采用固定阈值 γ 对于阈值下限进行限制。从实验结果来看，本文提出的伪标签产生策略具有良好的扩展性，可以更好地对抗长尾分布现象，并充分挖掘未标注数据所蕴含的信息。

表 4 不同伪标签产生策略之间的对比

实验编号	α	γ	线上 F1
1	80	0.7	**0.6288**
2	0	0.95	0.6071
3	80	0	0.6112

3.2.3 训练策略

除 FGM 外，我们也尝试了多种其他的对抗训练策略，同时对损失函数进行重加权等尝试，以期更好地适应本次任务特性。表 5 对应的实验在 NVIDIA GeForce RTX 2080 Ti * 2 上进行，预训练模型使用 chinese-macbert-base，batch_size 设置为 12，训练 40 轮，数据采用 train_0；表 6 对应的实验在 NVIDIA Tesla V100 32G * 2 上进行，预训练模型使用 ernie-3.0-base-zh，batch_size 设置为 32，训练 40 轮，数据采用 train_1。最终的实验结果表明 FGM 与 PGD 更适用于本次的小样本分类学习任务，然而由于 PGD 的迭代训练开销较大，我们只将其应用于伪标签生成阶段，在实验阶段我们放弃了对其进行进一步的探索。

表 5 不同训练策略之间的对比

baseline	fgm	pgd	rdrop	ema	warmup
0.5857	0.5949	**0.5995**	0.5668	0.5878	0.5782

表 6 不同训练策略之间的对比

baseline	fgm	cb	rfl	ntrfl	dbfl
0.6099	**0.6243**	0.6104	0.6177	0.6117	0.6066

3.2.4 尾部数据增强

针对类别不平衡的问题，我们采用了回译和简单文本数据增强两种方法进行缓解。实验结果表明，数据增强方法对于模型的预测效果有显著提升。实验在 NVIDIA Tesla V100 32G * 4 上进行，预训练模型使用 ernie-3.0-base-zh，batch_size 设置为 128，训练 40 轮，实验结果如表 7 所示：

表 7 尾部数据增强前后对比

数据	train_3	train_3	train_2	train_2
未增强	0.6137	0.6180	**0.6258**	0.6099
增强	**0.6260**	**0.6228**	0.6257	**0.6209**

*注：表格第 1 列与第 2 列、第 3 列与第 4 列实验结果为同一实验配置下 K 折交叉验证的不同折次的结果

3.2.5 预测头架构

Bert 相关预训练模型的输出由四部分组成，其中包括模型各层输出的隐藏状态和序列的第一个 token 的最后一层的隐藏状态（pooler output，是由线性层和 Tanh 激活函数进一步处理产生的）。其中模型最后一层输出的隐藏状态通常用于命

名实体识别，而 pooler output 的输出通常用于句子分类[7]。同时也有研究表明 BERT 编码了丰富的语言学层次信息。表层信息特征在底层网络，句法信息特征在中间层网络，语义信息特征在高层网络[8]。如果将模型的前几层和后几层进行平均池化，分类效果会更好。因此我们对不同的预测头架构进行了实验，实验结果表明目前使用的最后一层输出的隐藏状态+多层感知机的分类效果最好。实验结果如表 8 所示，实验均在 NVIDIA Tesla V100 32G * 4 上进行，预训练模型使用 ernie-3.0-base-zh，batch_size 设置为 128，训练 50 轮，数据采用 train_1。

表 8 不同预测头之间的对比

模型	线上 F1
1. last_hidden+MLP	**0.6148**
2. Concat last_four_hidden	0.6048
3. pooler_output	0.6061
4. pooler_output+MLP	0.6143
5. Average (last_four_hidden, first_four_hidden)	0.6018

致谢

感谢中国科学院大学高级人工智能课程介绍本次竞赛并鼓励我们参加，感谢 CCF BDCI 组委会和智慧芽公司的全体相关工作人员，感谢小队的 4 位同学在一个月内的辛苦付出，感谢开源的预训练模型库 huggingface，感谢提供硬件计算资源的两位外校同学，感谢家人的默默陪伴。

参考

[1] Devlin J, Chang M W, Lee K, et al. Bert: Pre-training of deep bidirectional transformers for language understanding[J]. arXiv preprint arXiv:1810.04805, 2018.

[2] Sohn K, Berthelot D, Carlini N, et al. Fixmatch: Simplifying semi-supervised learning with consistency and confidence[J]. Advances in neural information processing systems, 2020, 33: 596-608.

[3] Sun Y, Wang S, Feng S, et al. Ernie 3.0: Large-scale knowledge enhanced pre-training for language understanding and generation[J]. arXiv preprint arXiv:2107.02137, 2021.

[4] Miyato T, Dai A M, Goodfellow I. Adversarial training methods for semi-supervised text classification[J]. arXiv preprint arXiv:1605.07725, 2016.

[5] Wei C, Sohn K, Mellina C, et al. Crest: A class-rebalancing self-training framework for imbalanced semi-supervised learning[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2021: 10857-10866.

[6] Wei J, Zou K. 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. 2019: 6382-6388.

[7] Alammar, J (2018). The Illustrated Transformer [Blog post]. Retrieved from https://jalammar.github.io/illustrated-transformer/

[8] Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What Does BERT Learn about the Structure of Language?. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 3651–3657, Florence, Italy. Association for Computational Linguistics.