一、背景介绍包括：提出算法的背景、动机、解决了什么问题（需要介绍新的数据集或者基于课堂整理好的公开数据集，该部分不少于 2 页）（20 分）

随着互联网的迅速普及，信息的传播变得更加便捷和迅速，新浪微博、推特等社交媒体已成为人们获取新闻的重要来源，但与此同时假新闻的传播速度和广度达到了前所未有的高度。根据微博 2020 年度辟谣报告，全年有 76107 条虚假新闻被当局处理。由于假新闻不仅误导公众，造成社会恐慌，还可能引发政治动荡和经济损失，所以假新闻检测是一个需要解决的关键问题。

假新闻检测主要是通过自然语言处理和机器学习以及深度学习技术来识别和过滤虚假信息。传统的假新闻检测方法通常依赖于手工编写的规则和特征，通常是针对特定类型的假新闻和特定语言模式设计的，难以应对假新闻形式的多样性。而且人工特征选择过程往往依赖于专家的知识和经验，这种方法具有很大的主观性和局限性。专家可能无法全面捕捉假新闻的所有特征，尤其是那些隐蔽性强或具有高度复杂性的假新闻。一旦假新闻的生成方式或表现形式发生变化，这些规则可能会失效，导致检测效果大幅下降。

为了解决这一问题，人们提出了各种方法，其中大多数方法是利用深度学习集中在单域假新闻检测(SFND)上，例如政治、卫生。然而，在实践中 SFND 也存在着许多挑战。第一，基于单一领域的文本进行训练的模型，通常只能有效检测该领域的假新闻。当应用于其他领域时，由于不同领域之间的语言风格、内容主题和表达方式存在显著差异，模型的性能会显著下降，难以泛化到其他领域。第二，单一领域训练的模型高度依赖于该领域的训练数据质量和数量。对于某个领域，假新闻的数量可以是极其有限的。因此，基于如此不充分的单域数据，这些检测模型的性能并不令人满意。

在实际场景中，现实世界的新闻平台每天发布不同领域的各种新闻。因此，提出了多领域假新闻检测(MFND)，旨在构建一个能够跨越多个领域来解决数据稀疏问题并适应不同语境的假新闻检测系统，以应对不断变化的假新闻生成和传播形式，提高模型的灵活性和适应性，在各个领域更全面和准确地识别假新闻。

尽管多领域域假新闻检测（MFND）在理论上具有很大的潜力，但在实践中现有工作的 MFND 模型仍面临诸多挑战。首先，假新闻在不同领域的表现形式和特征各不相同，构建包含多领域的高质量假新闻数据集并非易事。多领域数据稀缺性问题不仅限制了模型的训练效果，也影响了模型在不同领域的泛化能力。其次，不同领域的假新闻在语言表达和语境上存在显著差异。数据分布会随 domain 领域的不同而不同，这被称为 domain shift。例如，军事新闻中最常用的词是海军、陆军等，而教育新闻中最常用的词是学生、大学、教师等。这种语言和语境的多样性增加了 MFND 模型的复杂性和检测难度,这就需要 MFND 检测模型具备很强的适应能力，能够在特征和标签分布发生显著变化时仍然保持高效。最后，不同领域的假新闻传播模式（propagation patterns）差异很大，一个在政治

领域表现良好的模型，可能无法有效检测娱乐或健康领域的假新闻，因为其传播渠道、速度、受众和形式都不相同。例如，政治领域的假新闻可能更多地依赖于社交媒体和新闻网站，而健康领域的假新闻可能通过博客、医疗论坛和健康类应用传播。

为了研究 MFND 更好的解决当前 MFND 所面临的挑战，作者构建了一个全面的数据集 Weibo21[1]，包含了科学、军事、教育、灾难、政治、卫生、金融、娱乐和社会等 9 个领域的新闻。每个域名都包含新闻内容、发布时间戳、对应的图片和评论。由于假新闻是为了经济或政治利益而故意制造的，它经常包含固执己见和煽动性的语言。利用新闻内容的语言特征来检测假新闻是合理的。时间戳和评论也包括在新闻中，因为时间戳可用于进行顺序分析。而评论可以提供辅助信号，特别是当帖子中包含的信息不多时。最终，微博 21 包含了来自 9 个不同 domain 的 4488 条假新闻和 4640 条真实新闻。

此外，作者还提出了一个简单而有效的多领域假新闻检测模型（CIKM2021），即 MDFEND[1]，它利用一个领域门来聚合混合专家提取的多个表示。将 MDFEND 模型与单域、混合域和多域方法进行了比较，验证了模型的有效性。具体解决了以下问题：

(1)构建了一个包含多领域的高质量假新闻数据集 Weibo21。该数据集涵盖从 2014 年 12 月到 2021 年 3 月的假新闻和真新闻。考虑到粒度的一致性和适当性，最终得出了九个领域：科学、军事、教育、灾难、政治、健康、金融、娱乐和社会。

(2）通过将 domain embedding 和 sentence embedding 都输入到门控机制中，利用领域门来聚合混合专家提取的多个表示，更好地建模了领域关系。通过一种柔和地更加细腻、连续的结合方式，而不是简单的二分法方式，结合多个领域比粗略地将领域共享特征和领域特定特征分离开来（EDDFN）效果更好。

（3）实验验证了混合域模型和多域模型通常比单域模型工作得更好，同时多域模型总体上比混合域模型表现得更好，这侧面印证构造额外的高质量的多领域数据对于虚假新闻检测的未来发展非常重要。

同时，基于课堂整理好的公开数据集，我们将两个数据集合并。默认 Integration-based Fake 中真新闻假新闻合成的内容为假新闻，Integration-based Legitimate 中真新闻和真新闻合成的内容为真新闻。得到/data/merge_whole.json。总共生成 8623（5926+2697）条数据。并根据原文的新闻分类标准使用 ChatGLM-6B 去判断每一条合成新闻数据的种类得到领域表示。最后训练模型去检测该新数据集的真假新闻准确率和 f1-score 等评估指标。

但是本文仍未研究 MFND 的实时性。假新闻的传播速度极快，因此检测系统需要具备高度的实时

性，以便及时发现和阻止假新闻的扩散。然而，现有的许多假新闻检测模型计算复杂度较高，难以满足实时性要求。这就需要未来在模型设计上进行优化，以提高检测速度和效率。

二、相关工作的优缺点总结（对应论文的相关工作部分，包括以往工作的概况，已有论文的改进地方，该部分不少于 1 页）(10 分)

早期的研究使用了手工特征[2，3]，例如 Oluwaseun Ajao 在 2019 年提出利用情绪有关的特征，Carlos Castillo 在 2021 年提出使用微博帖子中来自用户发布和转发的行为、帖子文本以及外部来源的引用等作为手工设计的特征去辅助自动检测假新闻和谣言，都得到了更高的检测准确率。 这种方法在可解释性和计算资源方面有其优势，但在应对复杂性和泛化能力上存在明显不足。

于是最近一些研究工作开始使用新颖的传播模式进行结构建模或神经网络建模，例如 Jing MA 等人[4]在 2017 年试图根据微博的传播结构提出一种基于核的方法，称为传播树核，该方法通过评估传播树结构之间的相似性来捕获区分不同类型谣言的高阶模式。Han Guo[5]等人在 2018 年提出一种新的结合社会信息的层次神经网络(HSA-BLSTM)，利用不同层次和社会背景的分层表征来检测谣言，通过注意机制将社会语境整合到网络中，从而将重要的语义信息引入到框架中，从而提高谣言检测的鲁棒性。同时由于尽早有效地发现谣言是至关重要的，Changhe Song 等人[6]在 2019 年提出了一种新的早期谣言检测模型——可信早期检测(CED)，该模型将所有转发到谣言候选人的帖子视为一个序列，将寻求一个早期的时间点来做出可信的预测。虽然这些模型能够识别和学习复杂的传播行为和关系，减少对手工特征设计的依赖，但这些模型可解释性差，往往只适用于单领域难以泛化到其他领域，同时高度依赖于该领域的训练数据质量和数量，需要消耗大量的计算资源。

另一些研究工作联合使用文本和视觉特征进行多模态建模。例如 Zhiwei Jin 等人[7]在 2017 年提出了一种新的带有注意机制的递归神经网络(att-RNN)来融合多模态特征，通过 LSTM 网络获得文本和社会语境的联合特征，从而产生可靠的融合分类。Peng Qi 等人[8]发现假新闻图像可能在物理和语义层面上与真实新闻图像具有明显不同的特征，这在频率域和像素域分别可以清楚地反映出来。于是在 2019 年提出一种融合频率域和像素域视觉信息的新型框架多域视觉神经网络(MVNN)来检测假新闻。Junxiao Xue 在 2021 年提出了一种多模态一致性神经网络(MCNN)[9]，考虑了多模态数据的一致性，并捕捉了社交媒体信息的整体特征，包括五个子网络:文本特征提取模块、视觉语义特征提取模块、视觉篡改特征提取模块、相似度测量模块和多模态融合模块。这些工作与单文本检测相比，都显著提高了虚假新闻检测的准确性。多模态建模的虚假新闻检测模型整合文本、图像、视频等多种模态，信息互补，能够增强模型鲁棒性。但需要大量计算资源，不同模态的数据融合和对齐复杂，在数据上也需要大量多模态标注数据，增加数据收集和标注成本。

考虑到多模态的信息互补能够增强模型鲁棒性，多领域(多任务)学习也能够提升模型泛化能力，使得模型适用于更多的领域。其主要思想是联合学习一组领域(任务)，这在许多应用中被证明是有效的[10-12]。这些研究侧重于用多种表征来捕捉不同任务之间的关系。而每个任务都通过相互联系得到加强，包括任务间的相关性差异。然而，这些多领域(多任务)框架不适合用于假新闻检测。因此，有必要设计一种适合于多领域虚假新闻检测 MFND 的有效方法。

当前针对多领域虚假新闻检测模型有较少研究工作。Wang 等人[13]采用极小极大博弈的思想来提取事件不变(领域不变)特征，但忽略了特定领域的特征。Amila Silva 等人[13]在 2021 年提出了联合保存领域特定知识和跨领域知识来检测不同领域的假新闻，但没有显式地充分利用领域信息。同时，针对用于假新闻识别的数据集，包括 Liar[15]、CoAID[16]、FakeHealth[17]、Twitter16[18]和微博[19,20]，但它们没有多域信息。FakeNewsNet[20]只包含 PolitiFact 和 GossipCop 两个域，这两个域不足以进行多域假新闻检测。因此，迫切需要一个合适的多领域假新闻数据集和简单高效的能够跨越多个领域来解决数据稀疏问题并适应不同语境的假新闻检测模型，以应对不断变化的假新闻生成和传播形式，提高模型的灵活性和适应性，在各个领域更全面和准确地识别假新闻。

三、提出的模型方法的解读，包括公式符号和等式含义的解读、算法流程的解读、实验环境的介绍、对比方法的选择等（对应论文的方法部分，该部分不少于 1 页）(20 分)

①作者构造了新的多领域数据集 Weibo21。该数据集涵盖 2014 年 12 月到 2021 年 3 月的假新闻和真新闻。对于每一条新闻，我们收集(1)新闻内容，(2)图片，(3)时间戳，(4)评论。最终得到 4488 条假新闻和 4640 条真实新闻。在数据收集之后，作者对新闻进行了分类。考虑到粒度的一致性和适当性，最终得出九个领域：科学、军事、教育、灾难、政治、健康、金融、娱乐和社会。然后，作者将所有的新闻片段注释到上述九个领域。为了确保注释的公正性，作者聘请 10 名专家手动注释新闻。

②多领域模型 MDFEND（CIKM2021）与单领域方法一样作者将多域假新闻检测问题归结为二分类问题。

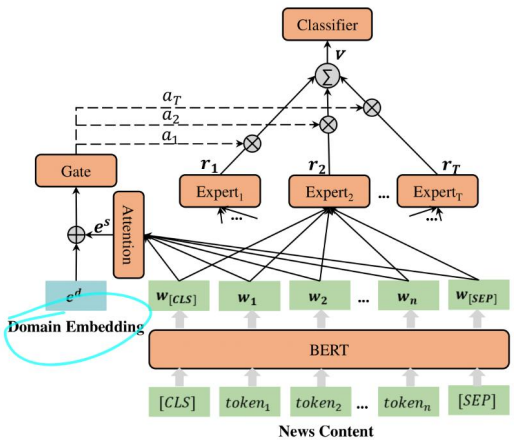


Figure 2: Overview of the proposed framework MDFEND.

(1) 首先对于一条新闻，使用 BertTokenizer 对其内容进行标记化。在添加了用于分类(即[CLS])和分隔(即[SEP])的特殊标记之后，获得了标记列表[[CLS],token1,token2,...,tokenn,[SEP]]，其中 n 是新闻内容中的标记（词）的数量。然后将这些标记送入 BERT 以获得词嵌入 $W = [w_{[CLS]}, w_1, ..., w_n, w_{[SEP]}]$。这些 word embedding 被送入了 a Mask-Attention network 以获得句子级别的 embedding e^s.

(2) 为了对每个领域进行特殊处理，定义了一个可学习向量 e^d，即领域嵌入，以帮助对每个领域进行个性化的表示提取。

(3) 雇佣了多个专家(即网络)来提取新闻的各种表示。直观地说，可以雇佣一位专家来提取多个领域的新闻表示。然而，一个专家只专注于一个领域，因此，单个专家提取的新闻表示只能包含部分信息，不能完全覆盖新闻内容的特征。因此，为了全面起见，作者聘请了多名专家。一个“专家”网络可以表示为Ψi(W; θi)(1≤i≤T)，其中 是作为“专家”网络的输入的一组单词嵌入，θi 表示要学习的参数，T 是指示专家网络的数量的超参数。让 r_i 表示“专家”网络的输出，即由相应的专家网络提取的表示。每个专家网络在本文中是一个 TextCNN[22] $r_i = \Psi_i(W; \theta_i),$

(4) 为了在 MFND 上获得良好的性能，需要生成能恰当地表示来自不同领域的新闻的高质量的新闻表示。直观地说，可以对所有专家的陈述进行平均。然而，简单的平均操作会去除特定领域的信息，因此合成表示可能不利于 MFND。请注意，不同的专家擅长不同的领域，他们擅长处理不同的领域。

因此，对于 MFND，作者适应性地选择专家，提出了一种以领域嵌入和句子嵌入为输入的领域门来指导选择过程选择过程的输出是表示每个专家的权重比的矢量α，表示不同专家重要性的权重向量。

$$a = softmax(G(e^d \oplus e^s; \phi)),$$

其中 G(∗;ϕ)是 domain gate,ϕ是 domain gate 的参数。

e^s 是 sentence embedding,e^d 是 domain embedding。

(5) 利用领域门，得到新闻的最终特征向量 v

$$v = \sum_{i=1}^{T} a_i r_i,$$

(6) 输出分类：新闻的最终特征向量被馈送到分类器，该分类器是具有 Softmax 输出层的多层感知

(MLP)网络，用于假新闻检测：$\hat{y} = softmax(MLP(v)),$

(7) 评估：Cross-Entropy

$$L = -\sum_{i=1}^{N}(y^i \log \hat{y}^i + (1 - y^i)\log(1 - \hat{y}^i))$$

③baseline 和对比方法

实验中有三种基线：(1)单域基线：TextCNN_Single、BiGRU_Single 和 BERT_Single；(2)混合域基线：TextCNN_ALL、BiGRU_ALL 和 BERT_ALL；(3)多域基线：EANN、MMOE、MOSE 和 EDDFN。在单域基线中，作者一次在一个单一域上对一个模型进行实验。最后一列的结果是前所有列的平均值。在混合域基线中，作者一次在所有域上用一个模型进行实验，并分别计算每个域的 F1 分数，而最后一列的结果不仅仅是单域基线中前一列的平均值，而是使用所有域的数据计算的。多域基线中使用 1 了的模型根据其结构将来自不同域的数据组合在一起。

Table 2: Multi-domain Fake News Detection Performance (f1-score)

model	Science	Military	Education	Accidents	Politics	Health	Finance	Entertainment	Society	All
TextCNN_single	0.7470	0.778	0.8882	0.8310	0.8694	0.9053	0.7909	0.8591	0.8727	0.8380
BiGRU_single	0.4876	0.7169	0.7067	0.7625	0.8477	0.8378	0.8109	0.8308	0.6067	0.7342
BERT_single	0.8192	0.7795	0.8136	0.7885	0.8188	0.8909	0.8464	0.8638	0.8242	0.8272
TextCNN_all	0.7254	0.8839	0.8362	0.8222	0.8561	0.8768	0.8638	0.8456	0.8540	0.8686
BiGRU_all	0.7269	0.8724	0.8138	0.7935	0.8356	0.8868	0.8291	0.8629	0.8485	0.8595
BERT_all	0.7777	0.9072	0.8331	0.8512	0.8366	0.9090	0.8735	0.8769	0.8577	0.8795
EANN	0.8225	0.9274	0.8624	0.8666	0.8705	0.9150	0.8710	0.8957	0.8877	0.8975
MMOE	**0.8755**	0.9112	0.8706	0.8770	0.8620	0.9364	0.8567	0.8886	0.8750	0.8947
MOSE	0.8502	0.8858	0.8815	0.8672	0.8808	0.9179	0.8672	0.8913	0.8729	0.8939
EDDFN	0.8186	0.9137	0.8676	0.8786	0.8478	0.9379	0.8636	0.8832	0.8689	0.8919
MDFEND	0.8301	**0.9389**	**0.8917**	**0.9003**	**0.8865**	**0.9400**	**0.8951**	**0.9066**	**0.8980**	**0.9137**

④实验环境的介绍

为了公平比较所有方法设置了相同的参数。这些模型中的 MLP 使用相同结构，具有一个 one dense layer(384 个 hidden units)。在所有实验中句子的最大长度为 170，单词嵌入向量的维度固定为 768(BERT)和 200(Word2vec)。实验使用 ADAM 优化器，搜索其学习速率从 1e-6 到 1e-2。所有方法的 mini-batch 均为 64。为了提高实验的可信度，实验进行了 10 次，并记录了 F1-score。

四、复现代码的实验结果进行展示,并且跑新的数据集或者基于课堂整理好的公开数据集，分析实验现象、消融实验，截图关键代码部分和结果指标等,将实现的代码和结果上传到 Github（该部分不少于 2 页）(40 分)

①源代码复现（github：https://github.com/ml-master/MDFEND）

（1）使用 BertTokenizer 对其内容进行标记化。

```
def word2input(texts, vocab_file, max_len):   max_len: 170    texts: ['【三星折叠屏原型机曝光：双屏设计/非柔性屏】网友@黎启i
    tokenizer = BertTokenizer(vocab_file=vocab_file)   tokenizer: BertTokenizer(name_or_path='', vocab_size=21128, model_ma
    # tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
    token_ids = []   token_ids: tensor([[ 101,  523,  676,  ...,    0,    0,    0],\n        [ 101,  523, 5543,  ...,    0,
    for i, text in enumerate(texts):    text: '【转发！公安民警围捕盗车团伙壮烈牺牲还有4名嫌犯在逃】今天凌晨1时许，兰州市公安局便衣侦查支
        token_ids.append(
            tokenizer.encode(text, max_length=max_len, add_special_tokens=True, padding='max_length',
                             truncation=True))
    token_ids = torch.tensor(token_ids) #[len,max_lenth]
    masks = torch.zeros(token_ids.shape)   masks: tensor([[1., 1., 1.,  ..., 0., 0., 0.],\n        [1., 1., 1.,  ..., 0., 0
    mask_token_id = tokenizer.pad_token_id   mask_token_id: 0
    for i, tokens in enumerate(token_ids): #[len,max_lenth]保留每条数据tokenizer有效部位记录mask   tokens: tensor([ 101,  523,
        masks[i] = (tokens != mask_token_id)
    return token_ids, masks
```

（2）训练参数

```
trainer = {Trainer} <models.mdfend.Trainer object at 0x000001DCDC80C9D0>
  bert = {str} './pretrained_model/chinese_roberta_wwm_base_ext_pytorch'
> category_dict = {dict: 9} {'科技': 0, '军事': 1, '教育考试': 2, '灾难事故': 3, '政治': 4, '医药健康': 5, '财经商业': 6, '文体娱乐': 7, '社会生活': 8}
  dropout = {float} 0.2
  early_stop = {int} 3
  emb_dim = {int} 768
  emb_type = {str} 'bert'
  epoches = {int} 50
> loss_weight = {list: 4} [1, 0.006, 0.009, 5e-05]
  lr = {float} 0.0005
> mlp_dims = {list: 1} [384]
  save_param_dir = {str} './param_model\\mdfend'
> test_loader = {DataLoader: 29} <torch.utils.data.dataloader.DataLoader object at 0x000001DCDCC37850>
> train_loader = {DataLoader: 86} <torch.utils.data.dataloader.DataLoader object at 0x000001DCDCC34400>
  use_cuda = {bool} True
> val_loader = {DataLoader: 29} <torch.utils.data.dataloader.DataLoader object at 0x000001DCB7643E80>
  weight_decay = {float} 5e-05
```

（3）将这些标记送入 BERT 以获得词嵌入

```
self.bert = BertModel.from_pretrained(bert).requires_grad_(False)

if self.emb_type == "bert":
    init_feature = self.bert(inputs, attention_mask = masks)[0] #[batch_size,token,bert_emb_dim]
```

（4）送入了 a Mask-Attention network 以获得句子级别的 embedding e^s

```
outputs = torch.matmul(scores, inputs).squeeze(1)

# scores[bs,1,max_len] inputs[bs,max_len,emd_dim] 相当于使用 scores 对 inputs 的不同位
```

置进行加权求和。

```
class MaskAttention(torch.nn.Module):
    """
    Compute attention layer
    """
    def __init__(self, input_shape):
        super(MaskAttention, self).__init__()
        self.attention_layer = torch.nn.Linear(input_shape, 1)

    def forward(self, inputs, mask=None):  mask: tensor([[1., 1., 1.,  ..., 0., 0., 0.],\n        [1., 1
        scores = self.attention_layer(inputs).view(-1, inputs.size(1))  scores: tensor([[[0.0139, 0.0163
        if mask is not None:
            scores = scores.masked_fill(mask == 0, float("-inf"))
        scores = torch.softmax(scores, dim=-1).unsqueeze(1)
        outputs = torch.matmul(scores, inputs).squeeze(1)
        # scores[bs,1,max_len] inputs[bs,max_len,emd_dim] 相当于使用 scores 对 inputs 的不同位置进行加权求和.
        return outputs, scores #outputs为这个batch内的这条数据被emd为emd_dim了 outputs[bs,emd_dim]
```

（5）为了对每个领域进行特殊处理，定义了一个可学习向量 e^d，即领域嵌入

self.domain_embedder = nn.Embedding(num_embeddings = self.domain_num, embedding_dim = emb_dim)；domain_embedding = self.domain_embedder(idxs).squeeze(1)

（6）Gate 门(超参 num_expert=5)

gate_input = torch.cat([domain_embedding, feature], dim = -1)

gate_value = self.gate(gate_input)

```
self.gate = nn.Sequential(nn.Linear(emb_dim * 2, mlp_dims[-1]),
                                    nn.ReLU(),
                                    nn.Linear(mlp_dims[-1], self.num_expert),
                                    nn.Softmax(dim = 1))
```

（7）专家网络：这个专家网络的定义展示了一种结构，其中包含多个 CNN 提取器模块。ModuleList 包含 5 个 cnn_extractor 模块，每个模块的索引从 0 到 4，代表 5 个专家网络。每个 cnn_extractor 模块包含一个 ModuleList，这个 ModuleList 中有 5 个 Conv1d 层。每个 Conv1d 层的输入通道数为 768，输出通道数为 64。不同的 Conv1d 层有不同的卷积核大小（kernel_size）：1、2、3、5 和 10。通过在每个 cnn_extractor 中使用不同大小的卷积核，网络可以从不同尺度捕获特征。每个 cnn_extractor 模块可以看作一个独立的特征提取器，专门处理某一方面的特征。

```
ModuleList(
  (0-4): 5 x cnn_extractor(
    (convs): ModuleList(
      (0): Conv1d(768, 64, kernel_size=(1,), stride=(1,))
      (1): Conv1d(768, 64, kernel_size=(2,), stride=(1,))
      (2): Conv1d(768, 64, kernel_size=(3,), stride=(1,))
      (3): Conv1d(768, 64, kernel_size=(5,), stride=(1,))
      (4): Conv1d(768, 64, kernel_size=(10,), stride=(1,))
    )
  )
)
```

feature_kernel = {1: 64, 2: 64, 3: 64, 5: 64, 10: 64}

```
class cnn_extractor(nn.Module):
    def __init__(self, feature_kernel, input_size):
        super(cnn_extractor, self).__init__()
        self.convs = torch.nn.ModuleList(
            [torch.nn.Conv1d(input_size, feature_num, kernel)
             for kernel, feature_num in feature_kernel.items()])
        input_shape = sum([feature_kernel[kernel] for kernel in feature_kernel])

    def forward(self, input_data):
        share_input_data = input_data.permute(0, 2, 1)
        feature = [conv(share_input_data) for conv in self.convs]
        feature = [torch.max_pool1d(f, f.shape[-1]) for f in feature]
        feature = torch.cat(feature, dim=1)
        feature = feature.view([-1, feature.shape[1]])
        return feature
```

（8）输出分类 label_pred = self.classifier(shared_feature)

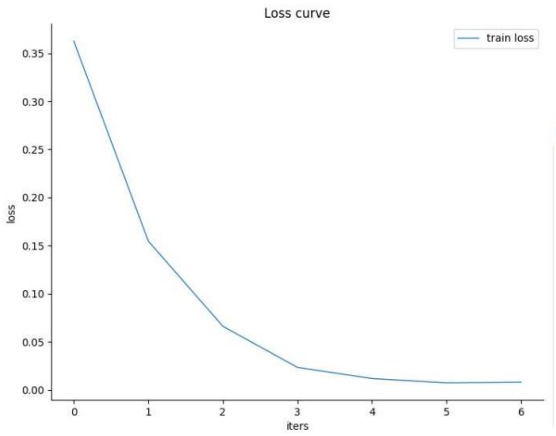
```
class MLP(torch.nn.Module):

    def __init__(self, input_dim, embed_dims, dropout, output_layer=True):
        super().__init__()
        layers = list()
        for embed_dim in embed_dims:
            layers.append(torch.nn.Linear(input_dim, embed_dim))
            layers.append(torch.nn.BatchNorm1d(embed_dim))
            layers.append(torch.nn.ReLU())
            layers.append(torch.nn.Dropout(p=dropout))
            input_dim = embed_dim
        if output_layer:
            layers.append(torch.nn.Linear(input_dim, 1))
        self.mlp = torch.nn.Sequential(*layers)
```

```
Sequential(
  (0): Linear(in_features=320, out_features=384, bias=True)
  (1): BatchNorm1d(384, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (2): ReLU()
  (3): Dropout(p=0.2, inplace=False)
  (4): Linear(in_features=384, out_features=1, bias=True)
)
```

（9）计算损失：

F.binary_cross_entropy(input, target, weight=self.weight, reduction=self.reduction)

设置了 early stop，并画出 loss 曲线




```
def judge(self):
    if self.cur['metric'] > self.max['metric']:
        self.max = self.cur
        self.maxindex = self.curindex
        self.showfinal()
        return 'save'
    self.showfinal()
    if self.curindex - self.maxindex >= self.early_step:
        return 'esc'
    else:
        return 'continue'
```

（10）展示结果：

Table 2: Multi-domain Fake News Detection Performance (f1-score)

model	Science	Military	Education	Accidents	Politics	Health	Finance	Entertainment	Society	All
TextCNN_single	0.7470	0.778	0.8882	0.8310	0.8694	0.9053	0.7909	0.8591	0.8727	0.8380
BiGRU_single	0.4876	0.7169	0.7067	0.7625	0.8477	0.8378	0.8109	0.8308	0.6067	0.7342
BERT_single	0.8192	0.7795	0.8136	0.7885	0.8188	0.8909	0.8464	0.8638	0.8242	0.8272
TextCNN_all	0.7254	0.8839	0.8362	0.8222	0.8561	0.8768	0.8638	0.8456	0.8540	0.8686
BiGRU_all	0.7269	0.8724	0.8138	0.7935	0.8356	0.8868	0.8291	0.8629	0.8485	0.8595
BERT_all	0.7777	0.9072	0.8331	0.8512	0.8366	0.9090	0.8735	0.8769	0.8577	0.8795
EANN	0.8225	0.9274	0.8624	0.8666	0.8705	0.9150	0.8710	0.8957	0.8877	0.8975
MMOE	**0.8755**	0.9112	0.8706	0.8770	0.8620	0.9364	0.8567	0.8886	0.8750	0.8947
MOSE	0.8502	0.8858	0.8815	0.8672	0.8808	0.9179	0.8672	0.8913	0.8729	0.8939
EDDFN	0.8186	0.9137	0.8676	0.8786	0.8478	0.9379	0.8636	0.8832	0.8689	0.8919
MDFEND	0.8301	**0.9389**	**0.8917**	**0.9003**	**0.8865**	**0.9400**	**0.8951**	**0.9066**	**0.8980**	**0.9137**

```
----------------------test_result--------------------
100%|                                                  | 29/29 [00:02<00:00, 12.42it/s]
{'科技': {'precision': 0.8182, 'recall': 0.6, 'fscore': 0.6923, 'auc': 0.899, 'acc': 0.8333}, '军事': {'precision': 0.9211, 'recall': 1.0, 'fscore': 0.9589, 'auc': 0.9731, 'acc': 0.9565}, '教育考试': {'precision': 0.8868, 'recall': 0.9216, 'fscore': 0.9038, 'auc': 0.9547, 'acc': 0.899}, '灾难事故': {'precision': 0.944, 'recall': 0.9672, 'fscore': 0.9555, 'auc': 0.9826, 'acc': 0.9295}, '政治': {'precision': 0.9055, 'recall': 0.9664, 'fscore': 0.935, 'auc': 0.9433, 'acc': 0.9064}, '医药健康': {'precision': 0.9417, 'recall': 0.951, 'fscore': 0.9463, 'auc': 0.9881, 'acc': 0.945}, '财经商业': {'precision': 0.8, 'recall': 0.8219, 'fscore': 0.8108, 'auc': 0.9674, 'acc': 0.8943}, '文体娱乐': {'precision': 0.8987, 'recall': 0.8452, 'fscore': 0.8712, 'auc': 0.9662, 'acc': 0.9271}, '社会生活': {'precision': 0.9069, 'recall': 0.91, 'fscore': 0.9085, 'auc': 0.9608, 'acc': 0.9007}, 'auc': 0.969594788429357, 'metric': 0.9119818819440524, 'recall': 0.912120487688262, 'precision': 0.9119092647924982, 'acc': 0.9120218579234972}
```

分析：文中 fscore 和 matric 记录的是不同领域的 f1score 和总数据的 f1-score。当超参数不同时，实验会使得领域内的数据会有细微的不同，但是总体的 ALL 的 f1-score 基本维持在 0.91。

论文中的实验或许是在不同的领域选择了不同的超参数，得到结果最好的那一个写入论文，但是总体的 matric（f1_score）基本维持在 0.91。

结果指标有：precision(精准度 / 查准率),accuracy(准确率),recall(召回率 / 查全率),f1-score= $$F1 = \frac{2 \times prediction \times recall}{prediction + recall}$$

②新数据集使用：Integration-based Fake（2697 条）和 Integration-based Legitimate（5926 条）

（1）【/data/gossip2merge.py】首先我们将两个数据集合并。默认 Integration-based Fake 中真新闻假新闻合成的内容为假新闻，Integration-based Legitimate 中真新闻和真新闻合成的内容为真新闻。得到/data/merge_whole.json。总共生成 8623（5926+2697）条数据。

（2）【/data/json2pkl.py】我们根据原文的新闻分类标准使用 ChatGLM-6B 去生成 merge_whole.json 每一条数据的种类得到领域表示。若是通过 llm 生成则得到/data/category.json。

Prompt 如下：prompt = "Please read the above paragraph and decide which topic in the folloing list ['Technology','Military','Education and Testing','Disaster incident', 'Politics', 'Medicine and Health', 'Financial Business', 'Entertainment', 'social life'] is the most relevant to the meaning of the above paragraph. You only need to choose the only one of the most relevant topic string name in the above list and answer one topic string name in the above list to output Please Don't print anything extra, no more than 3 words."

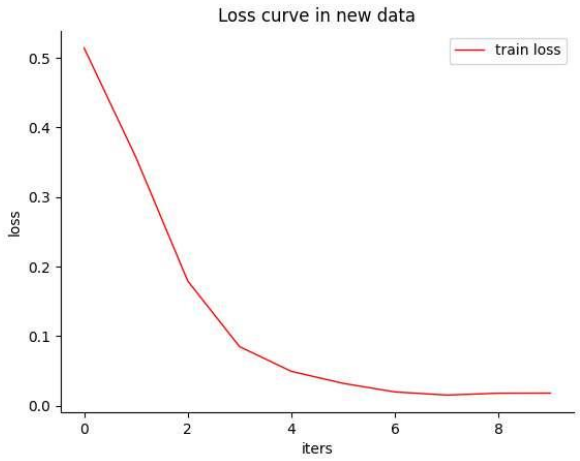
（3）通过生成的 /data/category.json 文件进行数据预处理为 category_change_no2entertainment.json 后再转化为/data/gossip 下的 pkl 文件后送入模型训练。

（4）训练

将原文适用于中文的 chinese_roberta_wwm_base_ext_pytorch 训练模型改为适用于英文的 bert-base-cased 训练模型获得词嵌入。

（5）loss 损失图（早停设置两个 epoch 的 loss 相差<0.001 停止，保存所有 epoch 里 test 测试集

真假新闻 accuracy 最高的模型）



（6）展示结果

```
--------------------------test_result--------------------
100%|██████████| 27/27 [00:07<00:00,  3.63it/s]
/root/miniconda3/lib/python3.10/site-packages/sklearn/metrics/_classification.py:1517: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 due to no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/root/miniconda3/lib/python3.10/site-packages/sklearn/metrics/_classification.py:1517: UndefinedMetricWarning: Recall is ill-defined and being set to 0.0 due to no true samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/root/miniconda3/lib/python3.10/site-packages/sklearn/metrics/_classification.py:1517: UndefinedMetricWarning: F-score is ill-defined and being set to 0.0 due to no true nor predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/root/miniconda3/lib/python3.10/site-packages/numpy/lib/function_base.py:520: RuntimeWarning: Mean of empty slice.
  avg = a.mean(axis, **keepdims_kw)
/root/miniconda3/lib/python3.10/site-packages/numpy/core/_methods.py:129: RuntimeWarning: invalid value encountered in scalar divide
  ret = ret.dtype.type(ret / rcount)
/root/miniconda3/lib/python3.10/site-packages/sklearn/metrics/_classification.py:1517: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 due to no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
{'Technology': {'precision': 0.9259, 'recall': 0.8065, 'fscore': 0.8621, 'auc': 0.8548, 'acc': 0.7714}, 'Military': {'precision': 1.0, 'recall': 1.0, 'fscore': 1.0, 'auc': 1.0, 'acc': 1.0}, 'Education and Test': {'precision': 0.0, 'recall': 0.0, 'fscore': 0.0, 'auc': 0, 'acc': nan}, 'Disaster incident': {'precision': 0.9091, 'recall': 0.7407, 'fscore': 0.8163, 'auc': 0.7778, 'acc': 0.7429}, 'Politics': {'precision': 0.8, 'recall': 0.5614, 'fscore': 0.6598, 'auc': 0.7327, 'acc': 0.6667}, 'Medicine and Health': {'precision': 0.8861, 'recall': 0.814, 'fscore': 0.8485, 'auc': 0.9013, 'acc': 0.8062}, 'Financial Business': {'precision': 0.5, 'recall': 0.2, 'fscore': 0.2857, 'auc': 0.5714, 'acc': 0.5833}, 'Entertainment': {'precision': 0.8739, 'recall': 0.7544, 'fscore': 0.8098, 'auc': 0.8477, 'acc': 0.7569}, 'Social Life': {'precision': 0.0, 'recall': 0.0, 'fscore': 0.0, 'auc': 0, 'acc': 0.0}, 'auc': 0.8443737896618501, 'metric': 0.7359213034968377, 'recall': 0.7582737474551865, 'precision': 0.7305257894875343, 'acc': 0.7546403712296984, 'legitimate_accuracy': 0.7482993197278912, 'fake_accuracy': 0.7682481751824818}
```

（7）分析：可能由于大模型生成的种类并不是很准确，同时该合成数据集的真假新闻数量差距大所以导致了模型不足以从其中捕获很好的检测真假新闻的能力。实验保存所有 epoch 里验证集真假新闻 accuracy 最高的模型，早停设置两个 epoch 的 loss 相差<0.001 停止。最终在测试集上面测试得到：在真新闻里预测准确率达到 74%，在假新闻里预测准确性达到 76%的结果。

五、参考文献(10 分)

[1] Nan Q, Cao J, Zhu Y, et al. MDFEND: Multi-domain fake news detection[C]//Proceedings of the 30th ACM International Conference on Information & Knowledge Management. 2021: 3343-3347.

[2] Oluwaseun Ajao, Deepayan Bhowmik, and Shahrzad Zargari. 2019. Sentiment Aware Fake News Detection on Online Social Networks. In ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2507–2511.

[3] Carlos Castillo, Marcelo Mendoza, and Barbara Poblete. 2011. Information Credibility on Twitter. In Proceedings ofthe 20th international conference on World wide web. 675–684.

[4] Han Guo, Juan Cao, Yazi Zhang, Junbo Guo, and Jintao Li. 2018. Rumor Detection with Hierarchical Social Attention Network. In Proceedings ofthe 27th ACM International Conference on Information and Knowledge Management. 943–951.

[5] Jing Ma, Wei Gao, and Kam-Fai Wong. 2017. Detect Rumors in Microblog Posts Using Propagation Structure via Kernel Learning. In Proceedings of the 55th Annual Meeting ofthe Association for Computational Linguistics (Volume 1: Long Papers). 708–717.

[6] Changhe Song, Cheng Yang, Huimin Chen, Cunchao Tu, Zhiyuan Liu, and Maosong Sun. 2019. CED: Credible Early Detection ofSocial Media Rumors. IEEE Transactions on Knowledge and Data Engineering (2019).

[7] Zhiwei Jin, Juan Cao, Han Guo, Yongdong Zhang, and Jiebo Luo. 2017. Multimodal Fusion with Recurrent Neural Networks for Rumor Detection on Microblogs. In Proceedings ofthe 25th ACM international conference on Multimedia. 795–816.

[8] Peng Qi, Juan Cao, Tianyun Yang, Junbo Guo, and Jintao Li. 2019. Exploiting Multi-domain Visual Information for Fake News Detection. In 2019 IEEE International Conference on Data Mining (ICDM). IEEE, 518–527.

[9] Xue J, Wang Y, Tian Y, et al. Detecting fake news by exploring the consistency of multimodal data[J]. Information Processing & Management, 2021, 58(5): 102610.

[10] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H Chi. 2018. Modeling Task Relationships in Multi-task Learning with Multi-gate Mixture-of-Experts. In Proceedings ofthe 24th ACMSIGKDD International Conference on Knowledge Discovery & Data Mining. 1930–1939.

[11] Zhen Qin, Yicheng Cheng, Zhe Zhao, Zhe Chen, Donald Metzler, and Jingzheng Qin. 2020. Multitask Mixture of Sequential Experts for User Activity Streams. In Proceedings ofthe 26th ACMSIGKDD International Conference on Knowledge Discovery & Data Mining. 3083–3091.

[12] Jiejie Zhao, Bowen Du, Leilei Sun, Fuzhen Zhuang, Weifeng Lv, and Hui Xiong. 2019. Multiple

Relational Attention Network for Multi-task Learning. In Proceedings ofthe 25th ACMSIGKDD International Conference on Knowledge Discovery & Data Mining. 1123–1131.

[13] Yaqing Wang, Fenglong Ma, Zhiwei Jin, Ye Yuan, Guangxu Xun, Kishlay Jha, Lu Su, and Jing Gao. 2018. EANN: Event Adversarial Neural Networks for Multi-Modal Fake News Detection. In Proceedings ofthe 24th acm sigkdd international conference on knowledge discovery & data mining. 849–857.

[14] Amila Silva, Ling Luo, Shanika Karunasekera, and Christopher Leckie. 2021. Embracing Domain Differences in Fake News: Cross-domain Fake News Detection using Multimodal Data. In Proceedings ofthe AAAIConference on Artificial Intelligence, Vol. 35. 557–565.

[15] William Yang Wang. 2017. “Liar, Liar Pants on Fire”: A New Benchmark Dataset for Fake News Detection. In Proceedings ofthe 55th Annual Meeting ofthe Association for Computational Linguistics (Volume 2: Short Papers). 422–426.

[16] Limeng Cui and Dongwon Lee. 2020. CoAID: COVID-19 Healthcare Misinformation Dataset. arXiv preprint arXiv:2006.00885 (2020).

[17] Enyan Dai, Yiwei Sun, and Suhang Wang. 2020. Ginger Cannot Cure Cancer:Battling Fake Health News with a Comprehensive Data Repository. In Proceedings ofthe International AAAIConference on Web and Social Media, Vol. 14. 853–862.

[18] Jing MA, Wei GAO, Prasenjit MITRA, Sejeong KWON, Bernard J JANSEN, KamFai WONG, and Meeyoung CHA. [n.d.]. Detecting rumors from microblogs with recurrent neural networks.(2016). In Proceedings ofthe 25th International Joint Conference on Artificial Intelligence (IJCAI 2016). 3818–3824.

[19] Jing MA, Wei GAO, Prasenjit MITRA, Sejeong KWON, Bernard J JANSEN, KamFai WONG, and Meeyoung CHA. [n.d.]. Detecting rumors from microblogs with recurrent neural networks.(2016). In Proceedings ofthe 25th International Joint Conference on Artificial Intelligence (IJCAI 2016). 3818–3824.

[20] Xueyao Zhang, Juan Cao, Xirong Li, Qiang Sheng, Lei Zhong, and Kai Shu. 2021. Mining Dual Emotion for Fake News Detection. In Proceedings ofthe Web Conference 2021. 3465–3476.

[21] Kai Shu, Deepak Mahudeswaran, Suhang Wang, Dongwon Lee, and Huan Liu. 2020. FakeNewsNet: A Data Repository with News Content, Social Context, and Spatiotemporal Information for Studying Fake News on Social Media. Big Data 8, 3 (2020), 171–188.

[22] Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In EMNLP. 1746–1751.