中国科学技术大学

学士学位论文

基于孪生网络的针对文本分类的数据增强技术

作者姓名： 范睿

学科专业： 计算机科学与技术学院

导师姓名： 连德富教授

完成时间： 二〇二一年五月二十三日

University of Science and Technology of China

A dissertation for bachelor's degree

Data Augmentation Techniques for Text Classification based on Siamese Network

Author: Fan Rui

Speciality: Computer Science and Technology

Supervisor: Prof. Lian Defu

Finished time: May 23, 2021

致　谢

回顾过去本科四年，我终于要在此刻画上句号，心中有很多感慨，还有很多人、很多事需要被感谢。

感谢连德富老师对我的指导，及时地给我指明前进的方向。我的毕业设计的完成过程充满坎坷，其中有不少失去方向的时候，每当我感到迷茫，连老师总会帮我把问题梳理清楚并帮我寻找一个解决方案，如果没有连老师对我的帮助和支持，我无法顺利地完成这项工作。

感谢陈衿学姐给我提供的宝贵意见。由于工位的相邻，当我遇到问题的时候第一个可以求助的对象就是陈衿学姐。在无数个我代码跑不通、结果不乐观的夜晚，我都能得到陈衿学姐的安慰和改进意见。学姐的存在像是提前清除了一些我前进路上的绊脚石，让我可以更加顺利、高效的完成工作。

感谢我的父母和我的妹妹，虽然只有每天打电话的时候可以聊聊天，但是和你们聊天像给我补充了十倍能量，让我能加速回血，再干一百年也可以。

还需要感谢曾经为我答疑解惑过的同学：黄旭，余磊，你们曾经给 debug 走投无路的我提供了希望；一直在我身边共同奋斗的舍友：全映菱、王晶、秦亚璇，你们像是马拉松最后阶段给选手陪跑的人，让我能够不断坚持；还有一直以来相互鼓励的人：王志远，你像我的“垃圾桶”，我总会把我内心的烦扰跟你说，但你从不会嫌弃我。

还需要感谢的是我在“光·遇”里认识的朋友们，薯条、汉堡、麦旋风和辣翅，虽然在毕业设计的完成过程中我大幅度减少了与你们的交流，但是你们的存在给了我无声的安慰，即使每天相处时间只有几分钟，在这几分钟内也让我的精神感到快乐。

最后需要感谢的是我自己，感谢我自己的坚持。过去四年的起起落落或许让我变得不再和四年前那个初出茅庐的小姑娘一样。这四年并不是一帆风顺的，我有心情畅快感到骄傲的时候，我也有心情低落自暴自弃的时候，但我很庆幸我没有在自信时变得自负，没有在低落中变得抑郁，虽然有的时候会偏离轨道，但我一路走来的大致路线还是向着更优秀的人迈进的。我不敢说以后我能有多么大的成就和财富，我只希望我能一直保持希望，不断前进，做我喜欢的，一生开心。

目 录

中文内容摘要

随着自然语言处理领域的发展，文本数据的数据增强技术也突飞猛进。不同学者使用不同方式来增强数据，达到即使只有少量的标注数据也能让模型很好地拟合样本，提高模型预测准确率的目的。本文提出了一种新的数据增强方法，先向原数据集中加入正负样本进行扩充，再使用孪生网络结构用扩充的无标签文本数据集对模型进行预训练后，再正式训练分类模型。经过在三种不同数据集上的实验，都证明经过预训练后的模型能够取得更高的平均准确率。另外，本文还讨论了扩充数据集时正负样本的数量对于预训练效果的影响，并提出了可以取得更好效果的参数组合。

关键词：中国科学技术大学；学位论文；数据增强；孪生网络；对比学习

Abstract

With the development of NLP, data augmentation of text data has also advanced a lot. Aiming at enhancing the precision of models using small data sets, different researchers use different tricks to do data augmentation. In this work, I present a new perspective on how to train models using small amount of data. First, expand the quantity by adding positive samples and negative samples to the original data set. Second, train the model on a siamese network using the previous unlabeled data set. Last, train the model using the original labeled data set. Tested on three different data sets, the method is proved to have higher accuracy on test sets. Moreover, I discuss about the effects of the number of positive and negative samples and propose a better combination of the numbers of two kinds of samples.

Key Words: University of Science and Technology of China (USTC); Thesis; Data Augmentation; Siamese Network; Contrastive Learning

第一章　引言

第一节　选题背景与意义

文本分类任务在自然语言处理 (NLP) 领域是最基本的任务之一。文本分类问题有许多的表现形式，如情感分析、问答匹配、意图识别等等。不论在那种子问题下，只要涉及到深度学习，就必然要考虑数据集的问题。然而需要承认的是，深度学习的一大弱点就在于它需要大量的带标签的数据集来进行训练，模型的效果和数据集的大小直接挂钩。然而对样本进行标注的工作是费时、费力和耗钱的。因此自从深度学习被提出以来，很多学者也开始进行如何使少量数据更好地训练模型的研究。大部分学者的思路在于通过少量样本生成新的样本来增多数据集中样本的个数去解决这个问题[1-5]，一部分学者从无监督的角度去探索提升模型性能的可能[6]，另一些学者使用半监督（SSL)[7-9] 的方法来更有效的利用无标签的数据。在本篇论文中，我通过对数据集使用已有的一个方法进行扩充后，无标签地在孪生网络结构上预训练分类模型的方法令模型获得对数据集进行大致分类的知识，然后再在原有数据集上进行正式的分类训练。我在三种不同的二分类数据集上进行了实验，实验结果表明通过使用新数据无标签地预训练模型能够从一定程度上对模型的分类效果进行了提升。我的主要贡献如下:

- 提出使用对比学习的方法在孪生网络上无监督地对模型进行预训练，有效地提升了模型的准确率。
- 提出一种正负样本数量的组合来使预训练取得更好的效果。

第二节　论文结构

第四章主要进行了问题分析和介绍了相关工作；第五章首先介绍了数据集、模型结构和实验环境，然后详细介绍了整个训练的过程，最后点出了评价指标；第六章对实验结果进行了展示和分析，然后讨论了正负样本的个数的选择对模型效果的影响，最后分析了本文介绍的数据增强方法的局限性；第七章分析了本文工作的不足以及未来可以尝试改进的方向。

第二章　问题分析与相关工作

在文本分类问题中，当只有少量数据时，如何使模型更好地拟合数据，使数据的表达能力更强？Wei 提出 EDA 技术[1]，采用四种方式来扩充数据集；Yu 等人使用 BackTranslation 的方法[2]，通过句子翻译来扩充数据集；Wang 等人[3] 采用结合外部信息的方法，在单词层面进行替换来使数据集规模增大，除此之外，也有一些利用无监督学习的数据增强工作被提出，如 Xie 等人[6] 提出的 UDA 方法在训练过程中加入了无标签的数据，极大的提升了网络对数据集的拟合效果。但是，能不能有一些新的方法能够增强数据的表达能力呢？在 CV 领域中，存在利用孪生网络结构对模型进行更新的任务，其中 Raia 等人[10] 利用孪生网络结构完成了将高维数据降维的工作，成功地保证了在高维中相似的矩阵在低维中也相似。下面我会一一对他们的工作进行介绍。

第一节　数据增强

NLP 领域的数据增强技术已经有了许多工作。Wei 等人提出的 EDA[1] 取得了很好的效果。EDA 全称为 Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks。它选用四种方式由原始数据集生成新的数据，分别为同义词替换 (SR)、随机插入 (RI)、随机交换 (RS)、随机删除 (RD)。同义词替换的做法是：排除停用词，在一个样本中随机抽取 n 个单词进行同义词替换；随机插入的做法是：排除停用词，在句子中抽取一个词，从此词的同义词列表中选取一个同义词插入它的后面；随机交换的做法是在句子中随机选两个单词进行位置交换，重复 n 遍；随机删除的做法是以 p 的概率随机删除句子中的每一个单词。用这四种方法进行增强后的结果十分可观，在使用 100% 的原数据集进行训练的平均正确率是 88.3%，但是使用 EDA 进行增强后的数据只用 50% 的数据就可以达到 88.6% 的正确率。与此同时，使用 EDA 增强得到的新数据并没有丢失可信度，作者在将原数据与增强数据同时进行可视化后发现，增强数据全部都是围绕着各自源自的类别，并没有发生数据集被打乱的现象，类与类之间的界限十分鲜明，这也是我选择将 EDA 中的同义词替换方法运用到我的模型中的原因，使用 EDA 增强出来的结果不会打乱数据集，能够更有效地指导我的模型更新，详情请见第三章。

另一个在 NLP 数据增强方向应用的比较广泛的方法是 Yu 等人[2] 提出的回

译法 (Back Translation)。回译法的基本原理是通过将一个英文句子翻译成另一种语言，再翻译回英文就可以得到一个新的语义相同但表达不同的英文句子。在这个工作中，研究者使用了 NMT 模型来进行翻译工作，这是一个被证明已经十分成熟的翻译模型。作者使用了两种语言进行翻译，分别是英语和德语互译与英语和法语互译。回译法的效果在文章中也被提到，使用经过回译增强的数据集进行 QANet 的训练后再 Dev set 和 Teset set 上的 EM 和 F1 分数高于所有 baseline，且高于使用没有增强的数据训练的 QANet 模型。

此外, Wang 等人[3] 通过结合外部信息来生成新的数据样本。具体做法为使用 Word2Vec 生成词汇表中每个单词后, 对一个句子中的每一个词, 在词汇表中选择一个嵌入向量与它最接近的单词将它替换, 判断接近的评价指标为两嵌入向量的余弦相似度, 这样就可以得到与原句意思一致但表达不同的句子, 如”Being late is terrible”，将每一个单词用余弦相似度最接近的词替换之后得到了”Be behind are bad”。作者在此工作中除了使用空格分词外，另外采取了按照语义分词的方法，具体做法是使用 SEMAFOR 网络对 3,800,000 推特的推文来进行语义分词，然后将语义的分词结果训练 Word2Vec，生成每个语义框的嵌入向量，再按照上述方法进行数据增强，即对于句子中的每个词，选一个最相似推特中的语义框来进行替换。本文使用的数据集来自于推特 #petpeeve 标签下的推文，共有 3375 条数据，60 个分类。数据增强带来了很大的数据提升，在使用 Google News lexical 嵌入进行数据增强的实验中，F1 分数相较于无数据增强的单词级 baseline 提升了 6.1%，对于使用语义分词的数据增强相较于无数据增强的所有特征 baseline 提升了 3.8%（单词级 baseline 指的是在对每一条推文提取几个单词作为特征，而所有特征 baseline 指的是，作者除了提取单词特征，还提取了一些别的特征，比如词性特征、语义依赖元组、语义特征，这些都有成熟的模型可以做到。）

最后，又有人提出可以使用神经网络来对数据进行增强。Xie 等人[6] 在 2020 年发表的工作中提出可以用无监督学习来增强数据。方法的名称为 UDA，即 Unsupervised Data Augmentation。曾经也有人实现过，使用无监督的方法进行数据增强，他们的做法是将原数据输入到模型得到模型对输入数据标签的预测的分布 $p_\theta(y|x)$，然后对输入或者模型中间的隐藏层添加噪声 ϵ，可以得到一个加了噪声的预测分布 $p_\theta(y|x,\epsilon)$，通过将两分布的差异添加到损失函数中来增加对模型参数进行约束。但 UDA 的作者认为，可以将简单的添加噪声换成更强更可靠的数据增强方式，如 back translation、TF-IDF 单词替换等方法。使用这样的做法后，损失函数包含两部分，第一部分是正常有标签训练的 cross entropy 函数，另

一部分是原数据集和增强出来的数据集进过模型得到的两个标签的预测分布之间的差异，作者用一个参数 λ 来平衡这两部分在总损失函数中所占比例。实验结果显示，UDA 在 10% 的数据集和 100% 的数据集上训练都要比单纯地有监督学习的性能要高，在 10% 的数据集中，Top-1 和 Top-5 的正确率甚至高出了 13.69% 和 11.54%，这表明 UDA 不仅能增加数据的数量，还能有效利用无标签数据信息来对模型性能进行提升。

第二节　孪生网络

孪生网络结构目前在 CV 和 NLP 领域中有不少应用，通常会伴随自监督的训练方法来一并使用。它的模型结构如图 2.1。最基本的架构包含两个同样结构和参数的模型，由于有两个模型，相应的就有两个输入和两个输出。通常情况下，研究者会将两个相似（正样本）或者两个不相似（负样本）同时分别输入到两个模型中，将两个模型的输出拿到一起计算损失再进行反向传播来更新模型参数。由于此处两模型参数相同，实际上可以使用同一模型来依次接收两输入。当然，逐渐也衍生出了对孪生网络两个分支不同步更新的模型训练方法，此时就不能用一个模型进行实验。目前，孪生网络结构在对比学习、聚类等领域有着重要应用。

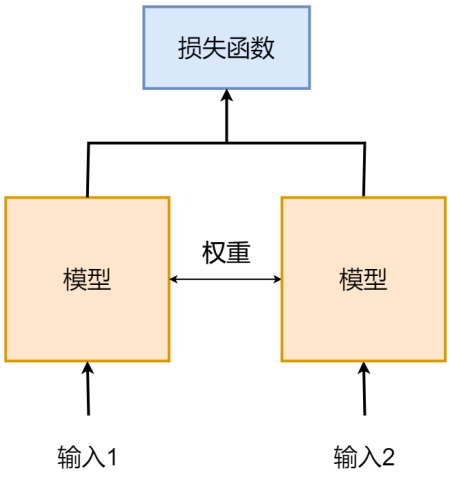

图 2.1　孪生网络结构

Raia 等人[10] 于 2006 年在 CVPR 中发表的一个将高维数据进行降维的工作利用了孪生网络的模型结构。该工作希望将一个高维数据通过模型计算得到一个低维数据，同时保证在低维的数据中进行分类时，对应的高维数据也能分类准确。为了达到该目标，若将模型从高维到低维的映射称为 G_w，G_w 需要满足以下

三个条件：(a) 输出空间内的简单距离衡量需要接近输入空间的邻里关系 (b) 对于没有近邻关系的样本，映射到的值应该是可信的 (c) 对于有扰乱的样本也可以正确地映射。作者采用了对比学习来实现这一想法：对于每一个高维数据，找到一个它的正样本和 K 个负样本，分 K+1 次同时将原高维数据和正、负样本输入进网络，通过对正、负样本定义不同的损失函数以达到“将原样本与正样本由原长为 0 的弹簧拉近，与负样本由原长为 m 的弹簧推远” 的效果。以下的损失函数可以实现：

$$L(W) = \sum_{i=1}^{P} l(W, (Y, \vec{X}_1, \vec{X}_2))$$

$$l(W, (Y, \vec{X}_1, \vec{X}_2)) = (1 - Y)\frac{1}{2}D_W^2 + Y\frac{1}{2}\{\max(0, m - D_W)\}^2$$

上式中 P 为样本集中样本对数；$\vec{X}_1, \vec{X}_2$ 为一次性输入到孪生网络两个分支的两个输入，令 $\vec{X}_1$ 为原高维数据，$\vec{X}_2$ 为另一个正或负样本；Y 表示 $\vec{X}_1$ 与 $\vec{X}_2$ 的关系，Y=0 表示 $\vec{X}_2$ 是与 $\vec{X}_1$ 相似的正样本，反之为不相似的负样本；W 表示模型参数，$D_W(\vec{X}_1, \vec{X}_2)$ 为两输入由模型得到的输出之间的距离，即 $D_W(\vec{X}_1, \vec{X}_2) = ||G_W(\vec{X}_1) - G_W(\vec{X}_2)||_2$，最后，m 为人为设定的参数。

本文的研究者使用 MNIST 数据集进行了实验。MNIST 数据集是一个手写数字的数据集，训练集包含 60,000 个样本，测试集包含 10,000 个样本。每个样本都是一个 28×28 大小的图片，每个像素点用一个灰度值表示。研究者在训练集中选取了 3,000 张 4 和 3,000 张 9，对于选取的每个样本计算 5 个欧式距离最接近的向量组成相似向量集合，总共组成 30,000 个相似对和 18,000,000 不相似对，由此构成新的训练集。由此训练得到的模型将 4 和 9 的高维数据映射得到的低维数据经过可视化表达后，可以发现有明显的区分。除此之外，研究者还对上述新构成的训练集加入了一些扰乱进行了实验，也验证了如果对训练集进行少量在水平或竖直方向上的平移，并不影响模型向低维数据映射的效果。最后研究者也用了 NORB 数据集验证了模型对同一物体的不同侧面映射的效果，感兴趣的可以移步原文进行进一步了解。

本方法所进行的数据增强工作是综合以上所有论文的实验得出的。如果可以用每一个样本生成与它相似的样本和不相似的样本作为正样本和负样本，并用孪生网络结构在正式训练模型前先无监督地对模型进行预训练，让模型具有一定的知识，可以判断对数据集进行一定程度的分类，那么在正式训练时会不会能够获得更高的准确率，怀着这样的想法，我设计了下面的模型并进行了实验。

第三章　数据与模型

第一节　数据集

本实验使用四个公开 nlp 分类数据集对模型的结果进行评估，分别是 IMDB 数据集（IMDB）、Subjectivity Dataset（Subj）、TREC 数据集和 PC 数据集。下面对此四个数据集进行详细介绍。

一、IMDB 数据集

IMDB 数据集是由 Maas, Andrew L. 和 Daly, Raymond E 等人于 2011 年在 ACL 中提出的[11]。全部数据集包含 50000 条电影评论，被平均分成了大小为 25000 的训练集和测试集。每条数据都有各自的二元情感标签（pos/neg）。数据的分布完全平均，即训练集和测试集中均有 12500 条 pos 数据和 12500 条 neg 数据。在整个数据集中，同一电影的评论不会多于 30 个，同时训练集和测试集来源的电影集合不相交，此举是为了防止模型通过记忆某一电影独特的特征来进行“作弊预测”。最后，数据集对 pos 数据和 neg 数据的划分标准为：当用户对电影的打分处于 [7,10] 间，为 pos；打分在 [0,4] 间，为 neg。那些处于 (4,7) 间的过于中立的用户评论没有被包含在数据集中。

本实验中只选取了训练和测试集的前 40% 的数据，即 5000 条 pos 和 5000 条 neg 进行训练和测试。实验中，pos 用 1 表示，neg 用 0 表示。

二、Subj 数据集

Subjectivity 数据集是由 Bo Pang 和 Lillian Lee 于 2004 年提出的[12]。数据集中含有 5000 条主观句子（或片段）和 5000 条客观句子。每条数据至少包含 10 和单词。主观句子来源于 Rotten Tomatoes 网站（http://www.rottentomatoes.com/）中的电影评论，客观句子来源于 IMDb 中对电影的情节介绍。需要了解的一点是 IMDb 中的介绍由于也是人工书写的，所以也有小概率会出现主观句子。

三、PC 数据集

Pros and Cons 数据集[13] 是由 Bing Liu 提出的二分类数据集。每一条数据都是用户对某一商品的评价：正面评价（如 Simple, Flexable, Reliable）被收录在 IntegratedPros 中，负面评价（如 Memory too small, slow charge between shots）收

录在 IntegratedCons 中。Pros 和 Cons 的数量相当。

四、CoLA 数据集

CoLA 数据集[14] 是由 Alex Warstadt 等人与 2018 年制作的文本二分类数据集。数据集的原始数据来源于 23 个语言学出版物，所有数据被分为两类：语法正确和语法错误。作者把所有句子分成了三个文件，分别是 in_domain_train.tsv,in_domain_dev.tsv,out_of_domain_dev.tsv。前两个文件有 in_domain 的前缀，意思是这两个文件里的数据都是来源于同样的 17 个出版物，而 out_of_domain 前缀的文件里的数据都是来源于另外 6 个出版物中。作者还分了两个文件夹，一个是 raw/，一个是 tokenized/。前者表示文件夹中的数据都是以原数据格式存储的，后者中存放的都是经过处理过的数据，比如将单词全部小写化等，作者是用 NLTK tokenizer 进行的处理。

数据集	训练集			测试集		
	类 1	类 2	总计	类 1	类 2	总计
IMDB	2500	2500	5000	2500	2500	5000
Subj	4000	3999	7999	999	999	1998
PC	18284	18190	36474	4475	4360	8835
CoLA	6023	2528	8551	354	162	516

Table 1：表中记录了各数据集中每个分类的具体数据条目数和总和数。注意表中的数据是训练中使用的样本数量，而不是数据集中真实的样本数量。

第二节　模型结构

本实验的主要模型由两部分组成：编码器和分类模型，模型结构图如图 3.2。下面依次介绍编码器和分类模型。

一、编码器

目前有许多性能可靠的编码器已经被提出，比如 BERT、谷歌的 word2vec、斯坦福的 Glove 等。通常情况下，编码器的主要工作是为了将一个单词映射到一个嵌入向量，同时保证嵌入向量能够完好的反应该单词的语义。在本实验中，我采用的编码器是 word2vec 中的 CBOW 模型。此模型计算嵌入向量的思想是：将

某一词当做中心词时，通过中心词上下文共 2c 个词来预测中心词。具体做法是：首先构建输入样本的词汇表，有了词汇表之后，每个词可以用各自的独热码来表示，若词汇表大小为 V，那么每个词的独热向量的维度为 $\mathbf{R}^V$；有了独热向量，通过依次预测每个单词来进行模型更新，假设此时想要预测第 i 个单词，首先取出第 i 个单词上下文各 c 个单词的独热向量，即 2c 个独热向量，将每一个向量与一个维度为 $\mathbf{R}^{V\times N}$ 的矩阵相乘，得到 2c 个 $\mathbf{R}^N$ 的向量，将此 2c 个向量取平均得到一个 $\mathbf{R}^N$ 大小的向量，得到隐藏层向量；将隐藏层向量再与一个维度为 $\mathbf{R}^{N\times V}$ 的矩阵相乘，重新得到了一个大小为 V 的向量，再将此向量进行 softmax 操作，得到了最终结果，此向量的每一位对应着词汇表中的每个单词在此上下文中出现的概率。此时，由于已知中心词，因此可以由 cross entropy 计算损失函数值进行模型的更新。

二、分类模型

分类模型也是来源于一个成熟的文本分类模型 TextCNN。TextCNN 利用卷积神经网络对文本进行分类，它包含四层，分别是卷积层、激活层、池化层和全连接层。

模型的输入是由编码器生成的一个句子的嵌入矩阵 ($\mathbf{R}^{N\times E}$)，其中 N 是句子长度，E 是嵌入大小（嵌入向量的长度），如图 3.2，假设一个 batch 中句子的最长长度为 500，嵌入向量维度为 300。

得到 500×300 的矩阵后，模型根据 kernel sizes 进行卷积层的计算。在这里，我将 kernel sizes 也做成了可以调节的参数，通常情况下，我会令 kernel sizes 为 [3,4,5]。根据 kernel sizes，模型会生成 9 个 kernel，分别是 3 个 3×300 的 kernel，3 个 4×300 的 kernel，3 个 5×300 的 kernel。每种 kernel 生成的数量与 kernel sizes 的长度一致，每个 kernel 的第一个维度大小是 kernel sizes 的每一位，第二个维度大小与嵌入向量的维度一致。卷积层的前向传播会生成 9 个一维向量，如图 3.2。

激活层使用了 ReLU 函数，ReLU 可以有效地避免梯度爆炸和梯度消失问题，能更有效的进行反向传播。

池化层接收激活层的输出，将每个卷积层输出的 9 个一维向量计算最大值，得到一个长度为 9 的的一维向量。

最后，将此长度为 9 的 1 维向量输入到全连接层，经过计算得到一个 2 维的输出向量作为模型的输出向量，输出向量的维度和数据集的标签数量一致。如果此时对此二维向量进行 softmax 操作，可以得到模型判定的此输入的类别，每一

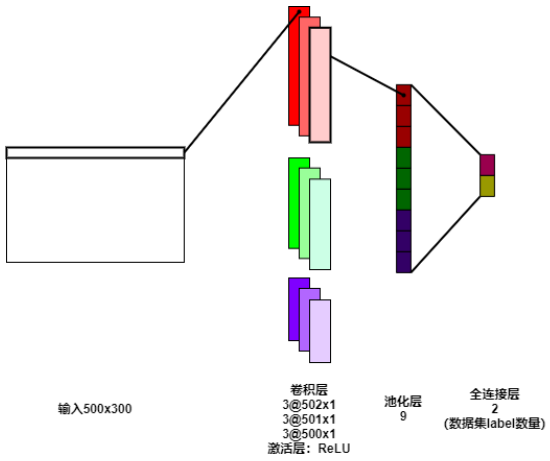

图 3.1　TextCNN 模型结构

位对应着此输入为这一位对应的类别的概率。不过这一步 softmax 操作不包括在模型内部，是属于模型输出后的结果处理的工作内容，因为在我的训练中，不是所有情况下都需要对模型的输出做 softmax 操作。

第三节　实验环境

系统型号为 x86_64 GNU/Linux，操作系统版本为 Ubuntu 16.04.1 LTS，内存 376.56GB，有两个 CPU，每个 CPU 有 20 核，处理器型号为 Intel(R) Xeon(R) Gold 6230 CPU @ 2.10GHz，GPU 型号为 GeForce RTX 2080Ti。

第四节　训练过程

模型的训练大致分为两步，如图 3.2。

一、Step1: 使用孪生网络结构预训练

首先，自监督地对模型进行预训练。所谓“自监督”指的是，不利用数据集的标签，只利用数据集中的原始数据来训练模型。具体做法借鉴第二章第二节中对比学习的工作：

利用 TextCNN 构造孪生网络的结构，即将孪生网络的两个分支的模型设置为 TextCNN 模型，且此两个模型共享所有权值，但有两个不同输入与不同输出。

对数据集中的每一个句子，找到它的 posNum 个相似样本和 negNum 个不相似样本，分 (posNum+negNum) 次同时送进模型，每次给一个分支输入原始数据，另一分支输入相似样本或其中一个不相似样本。相似样本构造方法：对句子进

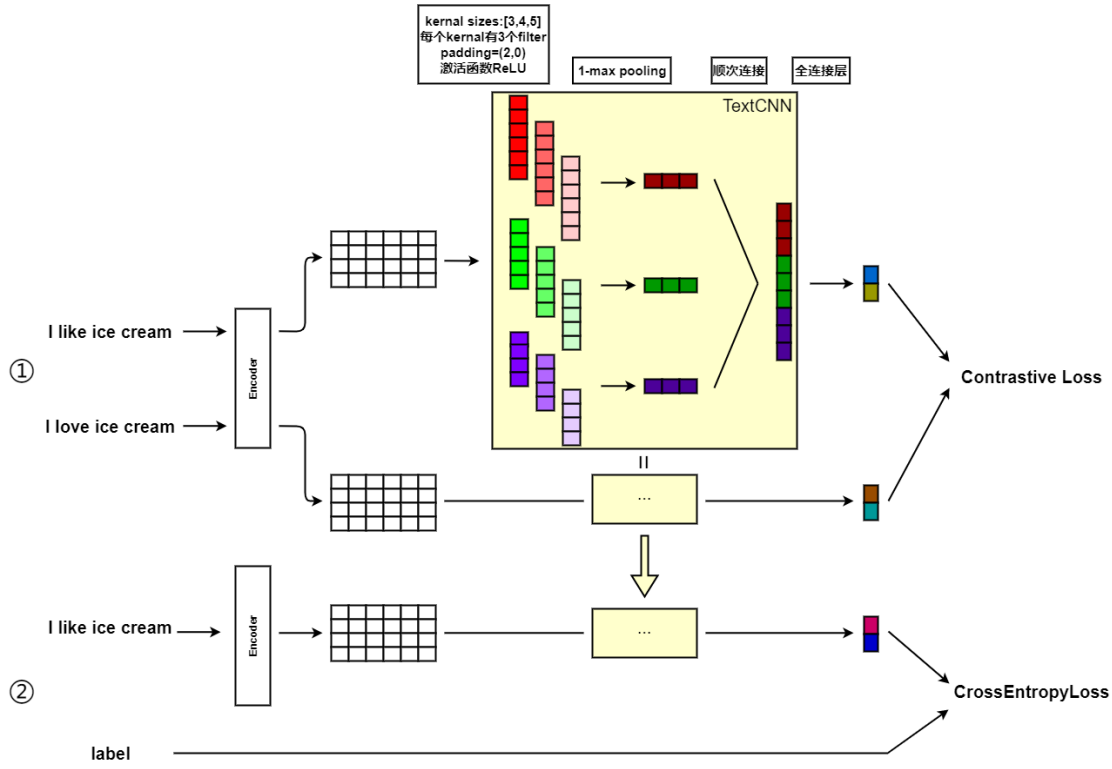

图 3.2　模型结构

行分词、词根化等处理后，随机选择一个单词进行同义词替换。同义词库选择的是 nltk.corpus.wordnet.synsets。不相似样本的构造方法：在数据集中随机选取 negNum 个句子作为负样本。对于 nltk.corpus.wordnet.synsets 这个同义词库需要注意几点：(a) 不是所有单词都有同义词，如 the,is 和一些专有名词。若没有对数据进行去停用词操作，会有很大概率遇到没有同义词的情况，我的处理是重新随机选一个单词进行同义词替换；若句子中没有词有同义词，则把这句话从数据集中删去；(b) 不是库中的所有词都和原词的语义完全相同。库中的词大部分是可信的，如 past 的同义词列表包括 yesteryear,last year 之类的，但也有少部分情况库中和原词在通常情况下语义毫不相关，如 sam 的同义词列表只包括 surface-to-air missile 和 SAM，前者是一种导弹，与原词的人名毫不相干，而后者在小写化后与原词完全相同。我目前没有避免前者这类语义不相关的情况，后者的情况我会识别出来，不会把 SAM 当做成一个有效的同义词。

得到由每一个句子生成的 posNum 个正样本和 negNum 个负样本组成的新数据集后，此时的数据还是文字而非嵌入矩阵，需要先用新数据集训练 word2vec 模型来得到每一个词的嵌入向量。word2vec 模型我是直接使用的 gensim 库中的 model.word2vec()，它可以直接根据输入的文本和参数进行 word2vec 模型的训练，训练结束后也可以直接根据单词索引嵌入向量。在我的实验中采用的参数

是：size=300，window=10，workers=10，iter=20。训练好模型后，读取每一个词的嵌入向量，将新数据集转化为嵌入矩阵，输入到孪生网络模型的两支里。

经过孪生网络后，对每一组输入都得到了两个输出。令两输入分别为 $\vec{X}_1, \vec{X}_2$，令 TextCNN 模型为 G_W，其中 W 为模型参数，那么两输出分别为 $G_W(\vec{X}_1), G_W(\vec{X}_2)$，由此可以计算两输出之间距离 $D_W(\vec{X}_1, \vec{X}_2) = ||G_W(\vec{X}_1) - G_W(\vec{X}_2)||_2$ 定义损失函数（Contrastive Loss）为：

$$L = \sum_{i=1}^{P} l(W, (Y_i, \vec{X}_i, \vec{X}_i'))$$

$$l(W, (Y, \vec{X}_1, \vec{X}_2)) = (1-Y)\frac{1}{2}D_W(\vec{X}_1, \vec{X}_2)^2 + Y\frac{1}{2}\{\max(0, D_W(\vec{X}_1, \vec{X}_2))\}^2$$

此损失函数是仿照第二章第二节对比学习的工作制定的，其中 P 是训练集中的数据对的数量，由于一条原数据可以扩展出 posNum 条正样本和 negNum 条负样本，所以 P 会是原数据集大小的 (posNum+negNum) 倍。损失函数的制定希望能将相似数据的距离拉近，将不相似数据的距离推远，此损失函数可以做到这一点的原因如下：

当 $\vec{X}_2$ 与 $\vec{X}_1$ 为相似样本时，$Y = 0$，$l = \frac{1}{2}D_W^2$，此时 l 对模型参数 W 的偏导为 $\frac{\partial l}{\partial W} = -\frac{\partial D_W}{\partial W}(-D_W)$，由此可以看出，当两输入为相似样本时，两输出的距离越大，损失函数对模型参数的更新力度越大，更新后距离更近，原长为 0 的弹簧将两向量拉近；相反的，当 $\vec{X}_2$ 与 $\vec{X}_1$ 为不相似样本时，$Y = 1$，$l = \frac{1}{2}\{\max(0, m - D_W)\}^2$，此时需要分两种情况来考虑：$D_W \geqslant m$ 时，$\frac{\partial l}{\partial W} = 0$，对参数没有更新作用，原长为 m 的弹簧没有将其拉近；$D_W < m$ 时，$\frac{\partial l}{\partial W} = -\frac{\partial D_W}{\partial W}(m - D_W)$，距离 D_W 在正数范围内离 m 约远，偏导越小，绝对值越大，对参数的更新力度越大，原长为 m 的弹簧希望将两样本距离推远。

参数 m 的选择需要靠经验调节，一般保证在训练开始前比样本集中的最远负样本对的距离要大就可以。

预训练工作的意义是，通过 contrastive loss 函数对模型参数的更新，模型知道了哪些样本是相似的，哪些样本是不相似的，虽然模型还不知道样本的真正标签，但是它已经可以将数据集分出个大概，这样后续的训练只需要告诉模型，它已经分出来的数据集哪一部分对应哪个标签，可以在速度和精度上都对模型的性能有提升。

二、step2: 使用带标签的原数据进行分类训练

对模型进行完预训练后，开始正式的模型分类训练。此时我们不需要之前计算的每一个样本的正负样本，我们只需要原数据集中的所有句子，并将所有

句子按照之前的 word2vec 模型中保存的结果转换成嵌入矩阵。将转化为嵌入矩阵的数据集依次输入到模型中，得到的输出再根据该输入的所属类别计算 Cross Entropy Loss 进行反向传播，进一步对模型参数进行更新。Cross Entropy Loss 的公式为

$$L(y_{true}, y_{pred}) = -\sum_i y_{true}(i)\log(y_{pred}(i))$$

其中 y_{true} 为真实的每个类别的概率, 即若某一数据类别为 0, 共有两个类别, 该数据的 $y_{true} = [1, 0]$; y_{pred} 为预测的个标签概率, 这里需要注意, 定义的模型输出向量并不是概率, 需要做一次 softmax 才能转化为概率的形式, 然而 pytorch 提供的 torch.nn.CrossEntropyLoss() 已经替我们做好了 softmax 工作，所以模型的输出可以直接传入 loss 函数中进行计算。最后, 若使用 pytorch 提供的 torch.nn.NLLLoss(), 需要在模型输出后进行 log_softmax() 操作，可以达到相同效果。

分类训练的意义在于，使已经能将数据集分类的模型能够直接给出具体的标签，由于此时模型已具有一定的知识，只要告诉模型一点标签数据，模型就可以举一反三地说出这一数据所属的集合中的全部数据的标签，所以即使使用少量数据进行分类训练, 经过预训练的模型的在 test 集上的准确率会比未经过预训练的模型高，这一结果在第四章中有体现。

第五节　评价指标

在本实验中，我是用 test 数据集上的正确率来评价模型的性能。由预测结果与真正结果，我们可以将数据集分成四个部分：TP,TN,FP,FN。TP 指真正为正例, 预测为正例的数据；TN 指真正为正例，预测为负例的数据；FP 指真正为负例, 预测为正例的数据；FN 指真正为负例，预测为负例的数据。在这种分类的前提下，由分类器预测正确的数据为 TP 的数据与 FN 的数据，因此，评价指标正确率的计算公式为:

$$Accuracy = \frac{TP+FN}{TP+TN+FP+FN}$$

第四章　结果与分析

本章会通过实验数据来说明经过数据增强并预训练后的模型能够在经过分类训练后取得更高的预测准确率。同时会探讨正负样本数对于预训练模型的影响。最后将会讨论使用本文方法进行预训练的局限性。

第一节　模型准确率

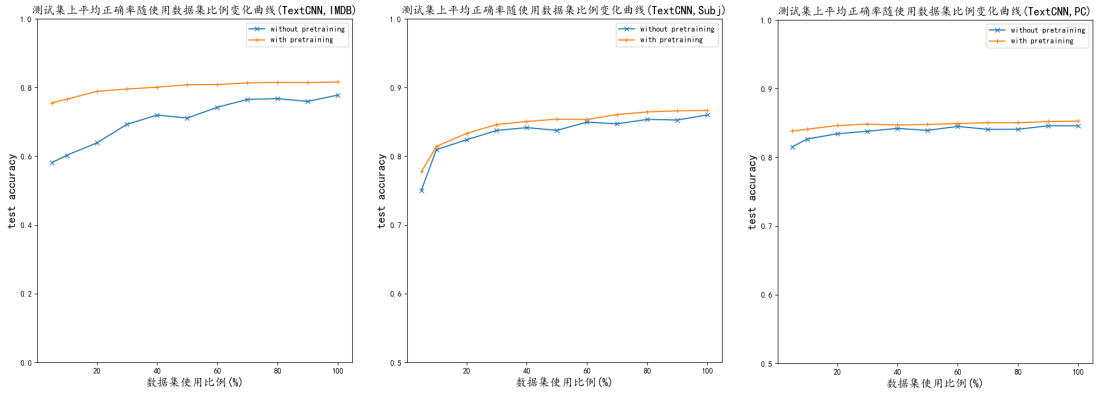

图 4.1　使用各数据集部分数据在有预训练和无预训练模型上的训练结果

数据集	无预训练	有预训练	性能提升 (%)
IMDB	0.7057	0.7985	13.2
Subj	0.8336	0.8449	1.40
PC	0.8382	0.8483	1.20

表 4.1：表中记录了在各数据集上有预训练和无预训练的平均正确率数值和预训练带来的效果提升。

训练过程：首先使用增强后的数据集对模型进行预训练，调整参数使得每个 epoch 模型在 test 数据集上正确率持续升高，到一定程度时结束预训练。这里并不能给出一个一致的标准认为预训练到哪一个程度就可以结束，有的数据集经过预训练可以达到 0.8 的正确率，有的去只能达到 0.6，但是在这两种结果的基础上进行后续的分类训练也都可以有不同程度的提升，只要预训练过程中模型在 test 集上正确率一直上升且模型逐渐收敛，就可以认为预训练结束。下一步再用原数据集的 5%，10%...90%,100% 在随机初始化的模型和经过预训练的模型上分别训练，记录每个 epoch 模型在 test 数据集上的正确率，最后将正确率在所有

epoch 上做平均，做出如图 4.1 的图像。

根据图像可以看出，随着使用数据集比例增高，两模型在 test 数据集上的表现都逐渐上升，同时，在不同数据集上，使用预训练的的模型正确率始终高于不使用预训练的模型，但也能看出，预训练带来的提升在不同数据集上表现不同，在 IMDB 数据集上表现最明显。还有一个明显的现象需要提出，在使用数据集较少的时候，有预训练的模型比无预训练的模型在 test 数据集上的准确率多出的部分要比使用数据集较多时多出的部分要多，这一现象在 IMDB 数据集上最明显，这也验证了第三章第四节训练过程 step2 中最后提出的结论：有了预训练后，使用少量数据进行分类训练就可以得到较好的分类结果，因为此时预训练的模型已经具备了足够的知识可以将数据集分成几个类别，只要分类训练在稍微给一点标签的信息，模型就可以举一反三地将样本所属的类别全部打上同一标签。

使用不同数据集进行训练时的各参数的值被记录在表 4.1 中：

数据集	预训练							分类训练				kernel sizes
	posNum	negNum	epoch	lr	l2	m	batch size	epoch	lr	l2	batch size	
IMDB	1	8	30	0.001	0.00001	0.7	128	50	0.0001	0.0001	64	[3,4,5]
Subj	3	5	20	0.0001	0.00001	1.0	64	50	0.0001	0.0001	64	[2,3,4,5]
PC	1	3	15	0.0001	0.00001	0.8	64	50	0.0001	0.0001	64	[2,3,4,5]

表 4.1：表中记录了使用不同数据集进行训练时的参数。其中 m 为预训练时认为设置的参数，详见第三章。posNum 与 negNum 指预训练对数据进行增强时，一个句子选取 posNum 个相似句子，negNum 个不相似句子

第二节　敏感性分析

在自监督学习的预训练中，数据增强得到的正样本和负样本的质量和数量至关重要。本节将讨论 posNum 与 negNum 对于预训练的影响。

一、负样本数量 negNum

图 4.2 画出了在固定 posNum=1 的情况下, 更改 negNum 的数值, 在 Subj 数据集上预训练 TextCNN 网络时每个 epoch 结束模型在 test 数据集上的表现。左边 (a) 图中的六个小图分别是在 negNum=2,4,5,6,10,12 的时候的数据，用不同 negNum 采集数据时，保证了训练模型的其他参数，如学习率、批大小等完全相同，右边的 (b) 图中画出了在不同 negNum 时，模型在 test 数据集上的平均表现。

由 (a) 中的 6 个小图可以看出，不是在 negNum 为任何值的时候自监督地对

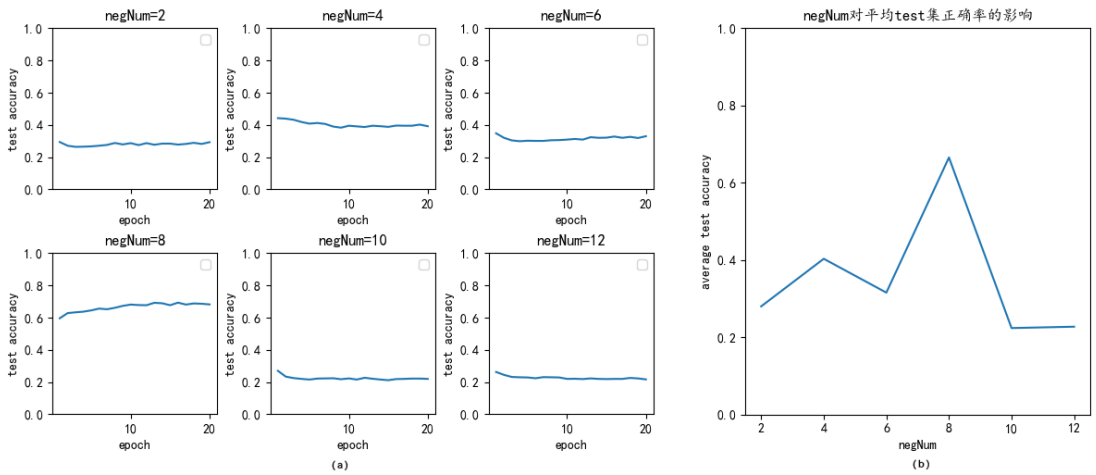

图 4.2　在 Subj 数据集上通过调整预训练参数 negNum 计算模型在测试集上的预测结果

TextCNN 做预训练都能够有助于模型在分类任务上的效果提高。当 negNum 为 2,6 和 8 时，模型在 test 集上的正确率随着 epoch 数的增多而升高，但为其他值的时候，test accuracy 会维持稳定甚至轻微下降。由 (b) 图可以看出，negNum 的选择并不是越多越好或者越少越好，随着 negNum 的增多，模型在 test 集上的平均正确率先呈升高趋势，再呈下降趋势，在 negNum 等于 8 时效果最好。猜想这一现象的原因是因为当 negNum 小时，选取的负样本少，没有过多的负样本数据能让模型参数将原数据与负样本的距离推远，对模型的更新力度不够，从直观上看，训练数据不够会导致模型训练效果较差是正常的事情；negNum 过大时，参与训练的样本过多，有可能出现过拟合的情况，所以在 test 集上准确率也不好。

二、正样本数量 posNum

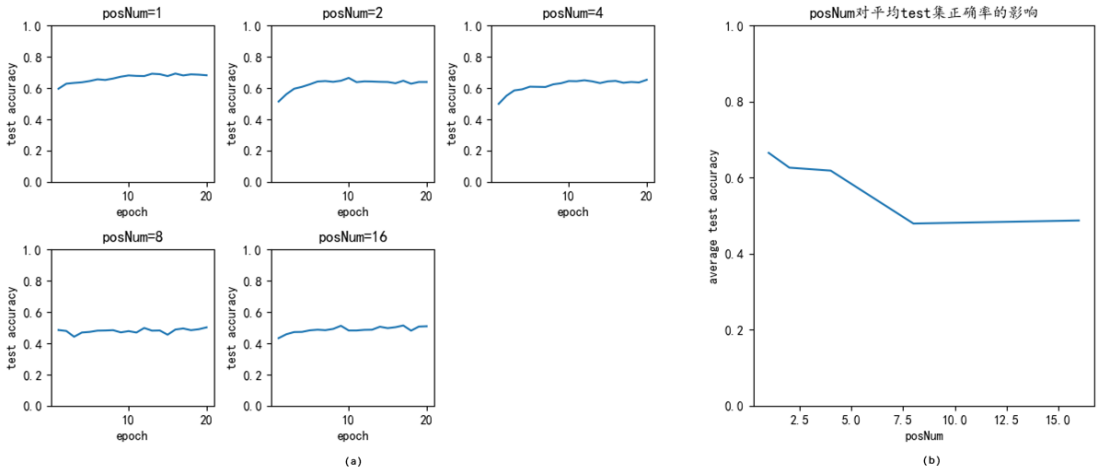

图 4.3　在 Subj 数据集上通过调整预训练参数 posNum 计算模型在测试集上的预测结果

图 4.3 画出了在固定 negNum=8 的情况下，更改 posNum 的数值，在 Subj 数

据集上预训练 TextCNN 网络是每个 epoch 结束模型在 test 数据集上的表现。左边 (a) 图中五个小图分别是在 posNum=1,2,4,8,16 时的数据，用不同 posNum 采集数据是，保证了训练模型的其他参数，如学习率、批大小等完全相同，右边的 (b) 图中画出了不同 posNum 时，模型在 test 数据集上的平均表现。

由 (a) 中五个小图可以看出，当 posNum=1,2 和 4 时，准确率随着 epoch 数的增加明显逐渐升高，当 posNum=8 和 16 时，准确率随着 epoch 的增加升高较缓慢；又 (b) 图可以看出，模型在 test 集上的平均准确率随着 posNum 的增大逐渐下降。这里我认为主要的原因在于同义词库的不可靠导致增强出来的正样本不全是对模型有指导意义的正样本，这一问题在第三章第四节中也有提到，使用的同义词库中的同义词不全是语义上相同的，当增强的正样本太多时，其实已经制造出了不少的扰乱样本对，这些样本对的存在不能对模型的预训练提供较好的指导作用，因此随着 posNum 的增加，模型的分类效果逐渐下降。

三、正负样本比例

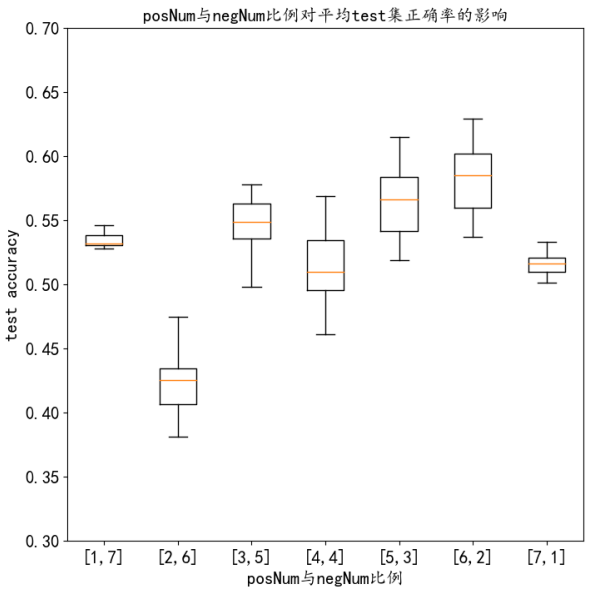

图 4.4　在 Subj 数据集上调整 posNum 与 negNum 的比例对 test 集正确率的影响

图 4.4 画出了在保证 posNum 与 negNum 总数不变（为 8）的情况下，改变 posNum 与 negNum 的比例，记录每个 epoch（共 20 个 epoch）模型在 test 集上的正确率画出的图像。其中每个小的箱线图，靠上的横线是最大值，靠下的图线为最小值，中间的方形上边为上四分位数，下边为下四分位数，中间的红线为中

位数。横轴为 posNum 与 negNum 的比例，左边为 posNum，右边为 negNum。从图中可以看出，当 posNum 与 negNum 偏差过大时，即 [1,7] 和 [7,1] 时，对应的箱线图很小，说明在预训练过程中，模型在 test 集合上的正确率变化很小；而当 posNum 与 negNum 比例相当时，即 [4,4] 周围，箱线图的跨度很大，说明个数相当的正负样本比例更有利于预训练的模型更新。通过观察还可以发现，当正负样本比例在 [6,2] 时，模型在 test 集合上的最高正确率、最低正确率费和平均正确率都是最高的，因此可以推断出当 posNum 是 negNum 的 1 到 3 倍时可以达到更好的预训练效果。

第三节　局限性

在实验过程中，我发现在使用 CoLA 数据集进行实验时，预训练始终不能取得一个可观的结果。经过原因分析，我认为是 CoLA 数据集的特殊性导致的这个问题。如第三章所述，CoLA 数据集的分类标准为句子是否为语法正确的，而与句子的语义和含义没有关系，在这样的情况下原来的正负样本的生成方法会造成错误：如果还使用同义词替换的方法来生成正样本，有可能会造成由一个语法正确的句子生成了语法错误的句子，因为同义词表中不是所有词都和原词的词性相同；如果还是用在数据集中随机选取句子作为负样本的方法生成负样本，有可能会造成对于语法不正确的句子选到的句子也是语法不正确的，应该被当成正样本的句子变成了负样本。之所以其他数据集可以用这样的方法生成正负样本的原因是它们被分类的标准都是各自的意思，而 CoLA 数据集分类的标准是语法的正确性，因此再使用这样的方法生成正负样本是不合适的。

第五章　总结与展望

本文提供了一种新的数据增强的思路，即先使用正负样本扩充数据集，然后无监督地预训练模型，再有监督地使用原数据集进行训练的方法，有效的对数据进行了增强。这样的方法在具有少量标注数据和大量未标注数据的数据集中十分适用。此方法在 IMDB 数据集上比无预训练的模型平均正确率提升了 13.2%。另外还探讨了正负样本的数量和比例对预训练效果的影响，并得出结论：正样本数是负样本数的 1-3 倍时可以获得更好的预训练结果。

在以后的工作中，可以尝试使用多分类的数据集对模型进行测试和改进，并且解决第四章提出的有关 CoLA 数据集的问题。

参 考 文 献

[1] Wei J, Zou K. EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks[J]. arXiv e-prints, 2019: arXiv:1901.11196.

[2] Yu A W, Dohan D, Luong M T, et al. QANet: Combining Local Convolution with Global Self-Attention for Reading Comprehension[J]. arXiv e-prints, 2018: arXiv:1804.09541.

[3] WANG W Y, YANG D. That's so annoying!!!: A lexical and frame-semantic embedding based data augmentation approach to automatic categorization of annoying behaviors using #petpeeve tweets[C/OL]//Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. Lisbon, Portugal: Association for Computational Linguistics, 2015: 2557-2563. https://www.aclweb.org/anthology/D15-1306. DOI: 10.18653/v1/D15-1306.

[4] Coulombe C. Text Data Augmentation Made Simple By Leveraging NLP Cloud APIs[J]. arXiv e-prints, 2018: arXiv:1812.04718.

[5] Guo H, Mao Y, Zhang R. Augmenting Data with Mixup for Sentence Classification: An Empirical Study[J]. arXiv e-prints, 2019: arXiv:1905.08941.

[6] Xie Q, Dai Z, Hovy E, et al. Unsupervised Data Augmentation for Consistency Training[J]. arXiv e-prints, 2019: arXiv:1904.12848.

[7] THOMAS P. Semi-supervised learning by olivier chapelle, bernhard schölkopf, and alexander zien (review)[J]. IEEE Transactions on Neural Networks, 2009, 20: 542.

[8] Bachman P, Alsharif O, Precup D. Learning with Pseudo-Ensembles[J]. arXiv e-prints, 2014: arXiv:1412.4864.

[9] Tarvainen A, Valpola H. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results[J]. arXiv e-prints, 2017: arXiv:1703.01780.

[10] RAIA H, SUMIT C, YANN L. Dimensionality reduction by learning an invariant mapping.[C]//In CVPR, 2006.

[11] MAAS A L, DALY R E, PHAM P T, et al. Learning word vectors for sentiment analysis[C/OL]//Proceedings of the 49th Annual Meeting of the Association for

Computational Linguistics: Human Language Technologies. Portland, Oregon, USA: Association for Computational Linguistics, 2011: 142-150. http://www.aclweb.org/anthology/P11-1015.

[12] PANG B, LEE L. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts[C]//Proceedings of the ACL. 2004.

[13] MURTHY G, BING L. Mining opinions in comparative sentences.[C]//COLING 2008: In Proceedings of the 22nd International Conference on Computational Linguistics: volume 1. Stroudsburg, PA, USA: 241-248.

[14] WARSTADT A, SINGH A, BOWMAN S R. Corpus of linguistic acceptability [Z]. 2018.