引　言

随着互联网、手机、社会媒体平台的普及，越来越多的人喜欢通过网络平台表达自己的看法和观点，对人物、事件、产品表达自己的意见及体验。在当今这个信息爆炸的时代，面对海量信息，用户在选择产品或服务的选择通常会参考他人的意见。而且，用户通过在线评论发表的感受，不仅仅是其他用户在产品和服务选择上的依据，更是商家了解产品服务的缺陷以及用户需求变化趋势的一个重要渠道，也是网络口碑的一种重要形式。毫无疑问的是，用户发表的评论包含着巨大的价值。用户主观发表的这些有价值的评论，包含着用户的各种情感，精确挖掘用户对产品、服务、事件的情感态度，是企业改进产品、消费者选择产品、政府舆情把控的有利的决策支持。然而，数据规模的爆炸式增长，使得信息消费、数据挖掘成为一项具有挑战性的任务。人工定位所需信息超出了任何人的能力范畴，不具可行性，因而机器辅助的文本处理便十分有价值，情感分析，也称为意见挖掘[1]，作为其中一项基本任务、一个自然语言处理领域的重要课题，更是备受关注。

情感分析可以分为三个层次：文档级、句子级、方面级。方面即主题，指评论的对象的实体属性。一个文档含有很多句子，一个句子可能含有多个不同的方面。文档级和句子级情感分析都是假设只含有一个主题，围绕整个主题分析，基于整个文档/句子，而非基于方面。例如。相比之下，基于方面的情感分析（Aspect Based Sentiment Analysis，ABSA）是一种细粒度的情感分析任务，旨在识别一条句子中一个指定方面的情感极性，对方面与上下文的结合让其尤为具有挑战性。情感极性即情感的类别，SemEval 在 2014 提出的任务将其分为正面、负面、中性三个类别。基于方面的情感分析共包含四个子任务，分别是：方面术语的情感分类（Aspect Term Sentiment Classification）、方面类别的情感分类（Aspect Category Sentiment Classification）、方面术语抽取（Aspect Term Extraction）和方面类别抽取（Aspect Category Extraction）。按对象划分可以划分为方面术语情感分析（Aspect Term Sentiment Analysis，ATSA）和方面类别情感分析（Aspect Category Sentiment Analysis，ACSA）两类。方面术语指在文本中出现的评论对象的词语序列，比如’The decor is not special at all but their amazing food makes up for it.’的方面术语有’decor’和’food’。方面类别指预先定义好的描述事物角度的类别词袋中的一个类别，比如’Love all their juices.’的一个可能的方面类别是’Drinks’。

2011 年，Jiang 等人提出了基于特征的支持向量机（Support Vector Machine，SVM）方法[2]。2014 年 Dong 等人提出了基于递归神经网络（Recursive Neural Network，RecNN）的方法[3]。尽管这些方法是有效的，但一方面通过人工特征工程手动设计一组依赖于方

面词的特征，是稀疏的，离散的，作为辅助信息来说是笨拙的，效果是十分有限的，另一方面，一般的神经网络结构具有高度的位置无关性，且需要序列填充到相同长度，因而不利于解决序列问题。早在 1990 年 Wojciech Zaremba 等人[4]就提出了循环神经网络（Recurrent Neural Network，RNN），解决了这一问题，然而受限于梯度消失与梯度爆炸，以及长期依赖中的灾难性遗忘，应用十分有限。1997 年 Hochreiter 等人[5]提出了长期短期记忆网络（Long Short-Term Memory，LSTM）在一定程度上解决了遗忘的问题，并且尽可能避免了梯度消失与梯度爆炸，然而依然受限于算力，并没有立刻得到广泛应用。而伴随着硬件能力的进一步发展，LSTM 逐渐成为了当时各种序列问题中模型选择的最优解。

2016 年，Tang 等人[6]在基于方面的情感分析领域首次对 LSTM 进行了巧妙的应用，他们提出了目标依赖的长期短期记忆网络（Target-Dependent LSTM，TD-LSTM）以及目标连接的长期短期记忆网络（Target-Connection LSTM，TC-LSTM）这两个方法，将方面词视作目标信息巧妙地融入 LSTM 中，从而显著地提高分类精度。对于 TD-LSTM，其基本思想是对目标字符串前后的上下文进行建模，这样两个方向的上下文都可以作为情感分类的特征表示，通过捕捉这种目标相关的上下文信息以提高目标相关的情感分类的准确性。具体来说，使用两个 LSTM 神经网络，左边一个 LSTM-L 和右边一个 LSTM-R，分别以左边和右边的上下文句子作为输入。我们从左到右运行 LSTM-L，从右到左运行 LSTM-R，然后连接 LSTML 和 LSTMR 的最后一个隐藏向量，并将它们馈送到 softmax 层以分类情感极性标签。也可以尝试对 LSTML 和 LSTMR 的最后一个隐藏向量进行平均或求和。考虑到 TD-LSTM 没有捕捉到目标词与其上下文之间的交互，基于这样的考虑，进一步提出了 TC-LSTM，其方法是将方面词从句中移除并且和句中其他每个词进行直接的拼接，其余部分不变，从而显式地利用目标单词和每个上下文单词之间的连接。TD-LSTM 和 TC-LSTM 取得了当时最先进的表现。之后，随着注意力机制的提出，Wang 等人[7]对注意力加以运用，提出了基于注意力与方面嵌入的 LSTM（Attention-based LSTM with Aspect Embedding，ATAE-LSTM）。在研究过程中，Wang 等人首先提出了带有方面嵌入的 LSTM（LSTM with Aspect Embedding，AE-LSTM）和基于注意力的 LSTM（Attention-based LSTM，AT-LSTM），然后对二者进一步实验、结合得到了 ATAE-LSTM。对于 AE-LSTM，其思想是，在对给定方面的句子进行极性分类时，方面信息是至关重要的，一句话中考虑不同方面十分可能会得到相反的极性，因此 Wang 等人首次提出了对方面词单独学习一个词嵌入向量，从而更好的利用方面词，其他词则是使用 Glove 预训练得到的词向量。对于 AT-LSTM，其做法是，对方面词和 LSTM 的每个时间步输出通过注意力机制的加性模型打分函数，来学习 LSTM 每个时间步输出对结果影响的权重，从而挑选出了句子

中对方面词有益的语境词。在 ATAE-LSTM 中，作者将二者结合，最终得到了当时最先进的表现，相比于 TD-LSTM，基于注意力机制让 ATAE-LSTM 从整个句子的 LSTM 输出中进行挑选，而不是仅从传递到最后一个位置的时间步来获取信息，这样也在一定程度上进一步克服了灾难性遗忘的问题。再后来，Sebastian Ruder 等人[8]提出了层次化的双向长期短期记忆网络（hierarchical bidirectional LSTM，H-LSTM）。Sebastian Ruder 等人表示过去的许多方法只能考虑句内的关系，并不能很好地捕捉句子之间的关系，而话语结构对情感预测有着十分重要的价值，于是提出了层次化的 LSTM。其做法是，首先对句子得到其词嵌入向量，然后将每个词向量输入到一个双向的 LSTM 中，作证称之为句子级的前向传播的 LSTM 和句子级的反向传播的 LSTM。然后句子的第一个词向量以及句子级的前向传播的 LSTM 的最后一个时间步的输出，和句子级的方向传播的 LSTM 的最后一个时间步的输出，这三个词向量输入到一个双向 LSTM 中，作者将其称之为观点级的前向传播的 LSTM 和观点级的反向传播的 LSTM，其时间步长度为 2，观点级的前向传播的 LSTM 和观点级的反向传播的 LSTM 的对应时间步的输出向量进行拼接，得到两个输出向量，作为观点级的双向 LSTM 的输出，然后将这两个输出进行拼接，输入到 softmax 输出层中从而得到最终的输出。H-LSTM 在当时也是取得了不错的表现，而其优点正如字面义，有着十分合理的层次化，这也为之后的研究提供了许多不一样的灵感，并增大了方法中模型架构的规模。

在此之后，以 LSTM 模型为基础，以层次化和注意力机制为主要手段，研究员们不断将过去的方法加以结合，或是按自己的想法调整架构，提出了各种新颖的方法，Liu 等人的基于注意力的双向 LSTM（BiLSTM-ATT-G）[9]，Chen 等人的循环记忆网络（RAM）[10] 等各种方法相继提出，又有孟霞对层次化双向 LSTM 进行了进一步尝试与调参优化[11]，对 LSTM 模型潜能的激发不断推向一个又一个高峰。在此期间，除了对 LSTM 的进一步尝试，还有对其他模型或机制的别具一格的应用。Xue 等人将计算机视觉领域的经验带入了自然语言处理领域，提出了带有方面嵌入的门控卷积神经网络（GCAE）[12]。GCAE 中，首先对句子做了词嵌入，然后对去掉方面词的上下文语境句子做了一维卷积，这里很巧妙的一点就是，具体来说作者实际上做了 n-gram，不同长度的 n-gram 得到的特征作为不同通道的输入，然后对卷积得到的特征分别通过门控 Tanh 单元（Gated Tanh Units，GTU）和门控线性单元（Gated Linear Units，GLU），结合方面词向量进行计算与激活，然后两个结果进行相乘，因而称作门控 Tanh-ReLU 单元（Gated Tanh-ReLU Units，GTRU），然后对这些输出做一个最大池化，输入到 softmax 层输出最终的情感分类。程康鑫基于研究者们对 LSTM 和 CNN 提出的灵感，提出了基于 LSTM 与 CNN 的中文餐饮评论情感特征提取算法[13]。Tang 等人提出了深度记忆网络（Deep Memory Network，

MemNet）[14]，该方法仅使用了注意力机制。在 MemNet 中，类似于 ATAE-LSTM，作者将方面词向量取出，如果方面词含有多个词，就求平均值。然后用语境词的嵌入矩阵和方面词间做加性模型，即一个对权重打分的小型网络，对方面词和语境词的拼接用 tanh 函数激活，然后用一个 softmax 输出层输出权重。纯注意力会导致位置信息丢失，这里作者尝试了四种特征工程的方法来加入位置信息。普遍来说，由多个层组成的计算模型能够学习多层次抽象的数据表示。作者表示一层注意力层本质上是一个加权平均合成函数，它不足以处理语言中的否定、强化和相反等复杂计算。因此作者设置了三层注意力层，认为通过足够多的这种变换的组合，可以学习对某一方面的句子表示的非常复杂的功能，然后最后一层输入到 softmax 输出层输出预测。MemNet 只使用了注意力机制，这让他训练很快，MemNet 这种注意力模型可以说是后来得到广泛应用的 Transformer 类模型的雏形，也是后来 Transformer 模型提出的灵感来源[15]。

2017 年，Ashish Vaswani 等人提出了 Transformer 模型[15]，该模型很快成为了继 MLP、CNN、RNN 后的第四大类模型。不同于 RNN 需要有时间步的 O（n）传递，Transformer 模型完全基于注意力机制，无需序列的传递，因而并行性十分强，训练速度很快，也因此没有灾难性遗忘的问题。多头注意力机制让 Transformer 能像 CNN 的多通道一样，学到多种形式的知识，键值对注意力实现了不同嵌入特征的序列转换，因而尤其适合作者提出时所做的翻译任务，在当时取得了最先进的表现，且训练速度十分得快。可惜的是原始的 Transformer 更加适合 seq2seq 类型的任务，在其他任务表现没有多大提升。而一系列基于 Transformer 的预训练模型的提出成功提高了整个自然语言处理领域表现的层次，其中 Jacob Devlin 等人结合了 ELMo[16]和 GPT[17]的思想而提出的 Transformers 的双向编码器表示（Bidirectional Encoder Representations from Transformer，BERT）[18]表现得尤为亮眼。在 BERT 提出之后，ABSA 领域中，很快便有了 Song 等人提出的 BERT-SPC 以及基于注意力编码网络的 BERT（Attentional Encoder Network-BERT，AEN-BERT）[19]，BERT-SPC 实际上是 BERT 的一个基线模型，提出者并未具体说明它的全称，方法是序列输入为[CLS]+上下文+[SEP]+方面词+[SEP]，从[CLS]位置得到编码输出，然后接入 softmax 层分类。AEN-BERT 中，输入分别是一个上下文句子的词嵌入和一个方面词的词嵌入，上下文输入是[CLS]+上下文+[SEP]，方面词输入是[CLS]+方面词+[SEP]，由 BERT 来做词嵌入，上下文嵌入矩阵做一个多头自注意力，另一边上下文嵌入矩阵和方面词嵌入矩阵间做一个多头注意力，这两个注意力输出的矩阵分别输入到一个对应位置的卷积中，或者说一个单层多感知机或全连接层，然后对这两个来自上下文的输出和方面词的输出，上下文的输出和方面词输出再做一个多头注意力得到打分权重，用这个权重和上下文的输出对应相乘，然后对上下文输出、加权的上下文输出、方面词输出这三

个输出做一个平均池化，然后用 softmax 得到分类。AEN-BERT 的亮点一是完全基于注意力机制，二是考虑到标签可能并不可靠，对训练集标签做了个 L2 正则化，这是非常新颖的想法。AEN-BERT 在当时也是取得了最先进的表现。Gao 等人仿照了 TD-LSTM 的思想提出了目标依赖的 BERT（Target-Dependent BERT，TD-BERT）[20]。虽然是借用了 TD-LSTM 的思想，TD-BERT 还是有很大不同的，它完美地体现了什么叫因地制宜。TD-BERT 给 BERT 的输入仿照的是 BERT-SPC，即[CLS]+上下文+[SEP]+方面词+[SEP]，然后 BERT 的输出，类似于 TD-LSTM，取出了方面词的输出，然后他们还取出了[CLS]的输出，充分利用了整句的编码，然后他们提出了两种对上下文和方面词间结合的方法，分别是对[CLS]的输出和方面词输出做点积，以及对[CLS]输出和方面词输出做拼接，然后对这样一个结合的结果全连接，得到最后的情感分类。TD-BERT 的架构看起来非常简单，但实验却证明了 TD-BERT 反而比 AEN-BERT 这种复杂的网络效果要好，一些传统的架构思想可能并不适合于 BERT，对 BERT 潜能的激发实际上还有很长的路要走。后来 Zeng 等人就提出了一个非常棒的架构，聚焦于局部语境的 BERT（Local Context Focus-BERT，LCF-BERT）[21]。他们提出了通过语义相对距离（Semantic-Relative Distance，SRD）来划分局部和全局语境，SRD 是一个超参数，词 i 的$SRD_i \leq SRD$时被视作局部语境词，SRD_i的计算方式用语言可以概括为，某个词到方面词最左沿和最右沿的距离中的较小者。对局部语境聚焦的方法，他们提出了语境特征动态掩码（Context Features Dynamic Mask，CDM）和语境特征动态加权（Context Features Dynamic Weighted，CDW），具体来说 CDM 就是不在局部语境范围内的向量直接替换为 0，CDW 就是不在局部语境范围内的向量进行加权，权重是这个向量到局部语境的距离除以句子总长。SRD 和 CDM/CDW 就是 LCF-BERT 的核心思想。LCF-BERT 中，对局部语境和全部语境分别先用 BERT 做了词嵌入，然后对局部语境嵌入矩阵做了 CDW/CDM，然后接着做多头自注意力，另一边全局语境嵌入矩阵直接做一个多头自注意力，然后局部和全局的这两个输出进行拼接在用一层多头自注意力进行计算然后用了一层池化层来得到最后的分类结果。LCF-BERT 在当时也是取得了最为先进的表现。此后各种基于 BERT 的方法如雨后春笋般不断被提出，Alexander Rietzler 等人提出了跨域适应的 BERT（Cross-Domain Adapted BERT，BERT-ADA）[22]，Minh Hieu Phan 等人提出了聚焦于局部语境句法的 BERT（Local Context Focus on Syntax-BERT，LCFS-BERT）[23]。在此期间也有王昆等人尝试了对过去的方法进一步优化，提出了基于文本筛选改进 BERT 的长文本方面级情感分析[24]。

本文对过去基于 LSTM、BERT 的一些解决基于方面的情感分类的主流方法进行了进一步的研究，提出了三个全新的架构方式，并进行了比较。也对 BERT 用领域语料继续预训练做了进一步的研究，本文的主要贡献如下：

（1） 对普通的 Transformer 模型进行了尝试，提出了其可能并不适合于情感分类或其他序列性较强的任务。但是将 LSTM 和 Transformer 加以结合，或许能够优缺互补。

（2） 提出了一种基于目标的非随机的特定 dropout 的方式，证明了在消除过拟合的同时，可以在一定程度上提高 ABSA 任务中多隐藏层方法的表现。

（3） 证明了任务内语料继续预训练 BERT 并不一定能带来更好的表现。

（4） 证明了领域内语料继续预训练 BERT 有很大可能性提高表现，话题内继续预训练不一定能带来更好的表现，合适的跨话题继续预训练更容易带来明显更好的表现，用于继续预训练的数据集的样本分布的广度可能有着很重要的影响。

本文的结构如下：

引言描述了 ABSA 的相关背景，按时间发展顺序，全面剖析了近 10 年来研究的进展，表明了在 ABSA 领域的方法还有很多提升的可能性，描述了本文的主要贡献和结构。

第一章介绍了本系统用到的各种技术，包括编程语言、开发用到的各种类库与框架，以及部分数学算法思想和优化算法的介绍等。

第二章对实验用到的数据集及其分布特性，以及本文主要依赖的过去的方法以及本文所提出的三个新方法进行了进一步的描述，并阐明了方法的思想。

第三章对实验进行了描述，并与过去的方法进行了比较与分析。

第四章对实验中一些具体的例子进行了进一步的分析，点明目前方法中依然有待提高的地方，以及局部语境的有效性。

第五章简单地描述了系统的实现，包括系统的结构与界面。

结论部分再次描述了本文带来的贡献，并基于研究结果给出了对未来研究的建议。

参考文献部分说明本文引用的文章来源。

1 相关技术介绍

1.1 python 语言介绍

作为目前最受欢迎的语言之一，python 是一个高层次的解释性脚本语言，有着许许多多优点。首先 python 有相对较少的关键字，程序块由缩进标识，代码简洁，语法明确，结构简单，说明文档也多样且清晰，随时都可以用 help 函数查看某个包的帮助信息，因而既易于学习，也易于阅读。其次 python 拥有着强大的交互性，可以从终端输入执行代码并获得结果，互动的测试让 python 易于完成对代码片段的调试。最后 numpy、pandas、matplotlib、PIL 等类库让 python 让计算与绘制图像变得轻松，尤为适合数据科学领域，pytorch、tensorflow 等框架让 python 毫无疑问地成为深度学习的首选语言，底层使用 C++ 实现，也让其有着十分不错的运行效率。

1.2 python 类库介绍

python 有着许多强大的第三方类库，以下介绍本文主要使用到的类库。

1.2.1 numpy 介绍

numpy 是 python 科学计算即 anaconda 中的一个基础包，其核心是 ndarray 数组对象，有着强大的索引与切片操作，拥有各种强大的矩阵计算功能，与之相关的许多循环操作都是由底层的 C 语言实现的，这避开了 python 中循环效率低下的代价。向量化的代码也让 numpy 更加简洁更加、易于阅读。提供的许多高级数学和其他类型操作的 API，比如离散傅立叶变换、基本线性代数，基本统计运算和随机模拟等等，让代码量大幅减少，更少的代码行也意味着让程序员更容易犯更少的错误。向量中的广播机制让 numpy 充分地利用了重复指令操作下的单指令多数据流，从而使得运行效率十分地快。

1.2.2 pytorch 介绍

Pytorch 是一个基于 python 的开源科学计算库，由 Facebook 人工智能研究小组开发。Torch 作为 pytorch 的前身，采用的是小众的编程语言 Lua，导致流行度不高。Pytorch 和 torch 的底层都是 C 语言，只是上层包装语言不同。相比于 numpy，Pytorch 提供了更加灵活和快速的深度学习研究平台，有着强大的自动求导系统，并且提供了强大的 GPU 加速功能。相比于其他一些主流框架以及过去的 tensorflow1.0，pytorch 以强制性和动态图的方式包含所有内容，计算图在运行中定义，这样使得 pytorch 的运行效率十分的高效。

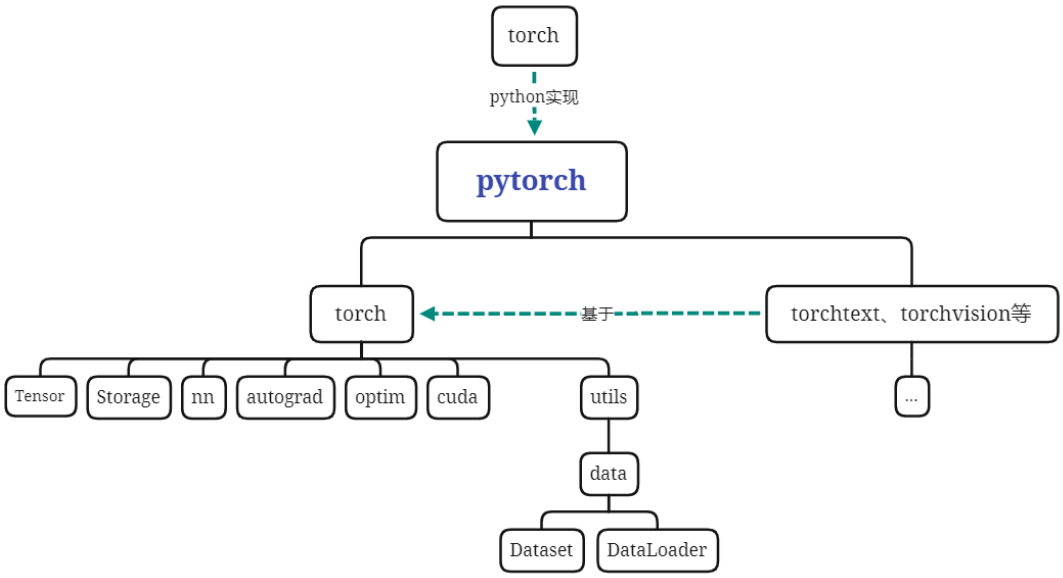

图 1.1　pytorch 结构图

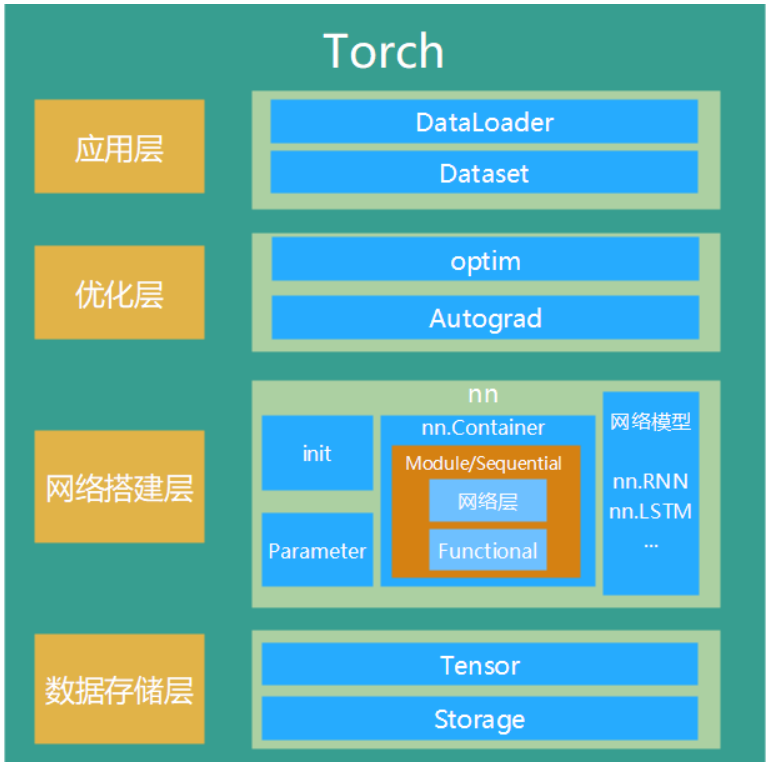

图 1.2　pytorch 架构图

用 Pytorch 框架编写一个深度学习模型的主要模块如下：

（1） 输入处理模块：将输入数据变成网络能够处理的 Tensor 类型，过程中我们一般会使用到 DataSet 来封装，DataLoader 来读入

（2） 模型构建模块：定义网络结构，目标是完成前向传播 forward 函数，使得网络能够从输入特征 X 通过前向传播得到一个预测的$\hat{y}$。

（3） 定义损失函数和优化器模块：通过对损失函数和梯度下降优化算法的定义，来实现自动求导以及反向传播更新网络参数。

（4） 构建训练过程：将上述模块按序组合起来，进行迭代训练。

1.2.3 Transformers 介绍

huggingface 是一家总部位于纽约的聊天机器人初创服务商，开发的应用在青少年中颇受欢迎，相比于其他公司，huggingface 更加注重产品带来的情感以及环境因素。但更令它广为人知的是 huggingface 对 NLP 技术的专注，拥有大型的开源社区。尤其是在 github 上开源的预训练模型库 Transformers，已被下载超过一百万次，超过两万次收藏。Transformers 提供了 NLP 领域大量时代最前沿的预训练语言模型结构的模型和调用框架。这个库最初的名称是 pytorch-pretrained-bert，它随着 BERT 一起应运而生。当时，BERT 以其强劲的性能，引起 NLP 领域研究者的广泛关注。它提供预训练模型的下载，使没有足够算力的开发者们也能够在几分钟内就实现最先进的微调。后来随 BERT，GPT，GPT-2，Transformer-XL，XLNET，XLM 在内六个预训练语言模型逐渐上传，pytorch-pretrained-bert 这个名字改成了 pytorch-transformers，势力范围扩大了不少。而后随着上百种语言，几十种预训练语言模型的不断上传，以及对 TensorfFlow2.0 和 pytorch 的双重支持以及可以在两者间随意迁移模型，其名称变作了 Transformers。Transformers 下的自然语言处理模型，简单、强大、高性能，既是新手入门的不二选择，也为研究者和开发者们提供了一个更高更快捷的平台。

1.3 最大似然估计

最大似然估计是一种重要而普遍的求估计量的方法。通俗易懂地来说，给定样本 X，我们可以计算出基于样本 X 发生某个事件的概率，这个概率同时受到某个参数θ的影响，随θ的变化而变化，但我们并不知道这个参数是多少，我们希望发生的概率尽可能大，固定一组特定采样$x_1, x_2, \ldots, x_n$，我们可以将θ视作自变量，P 视作函数，求得这个函数的最大值，从而得到一个估计的θ使得我们期望的概率 P 尽可能大。具体来说，可以描述成如下形式：

给定一概率分布 D，其概率密度函数为f_D，分布参数θ，从分布中抽出一个具有 n 个值的采样$x_1, x_2, \ldots, x_n$，通过概率密度函数我们就可以计算出其发生的概率：

$$P(x_1, x_2, \dots, x_n) = f_D(x_1, x_2, \dots, x\_n|\theta) \tag{1.1}$$

我们并不清楚θ的值，而希望其最大，于是此时将其看作对自变量θ的函数，定义可能性函数：

$$lik(\theta) = f_D(x_1, x_2, \dots, x\_n|\theta) \tag{1.2}$$

通过求导来求解θ的最值点从而得到lik(θ)函数的最大值，来使得 P 最大，得到最值点$\hat{\theta}$称作θ的一个最大似然估计。

更一般地，实际操作中，为方便计算，我们一般对log lik(θ)进行求导。

1.4 交叉熵损失函数

为了说明 softmax 回归，首先要说明其分类数为二的特殊情况，即 logistic 回归或称作 sigmoid 回归。逻辑回归中在一个样本特征 x 和特征参数w的条件下，输出的预测值$\hat{y}$表示预测标签 y 为$y = 1$的概率为$\hat{y}$，因此，预测正确的概率：

$$P(y|x) = \begin{cases} \hat{y}, y = 1 \\ 1 - \hat{y}, y = 0 \end{cases} \tag{1.3}$$

而带入一个分段函数求导计算最值并不是一个好的方式，上式可以通过合并得到如下公式：

$$P(y|x) = \hat{y}^y(1 - \hat{y})^{1-y} \tag{1.4}$$

根据极大似然估计，我们一般会选择求log P的最大值，易知 P 的范围在[0,1]，因此log P的范围在$(-\infty, 0]$，在机器学习领域，为便于观察，我们一般求−log P的最小值，这里的−log P正是逻辑回归中的损失函数：

$$L(\hat{y}, y) = -(y\log\hat{y} + (1 - y)\log(1 - \hat{y})) \tag{1.5}$$

softmax 回归中，设类别为 k，在一个样本特征 x 和特征参数w的条件下，输出的预测值$\hat{y}$是一个 k 维的列向量，$\hat{y} = [\hat{y}_1, \hat{y}_2, \dots \hat{y}_n]$表示预测$y_i = 1$的概率为$\hat{y}_i (i \in [1,2, \dots, k])$，即预测为第 i 类的概率为$\hat{y}_i$，因此预测正确的概率为：

$$P(y|x) = \prod_{i=1}^{k} \hat{y}_i^{y_i} \tag{1.6}$$

因此 softmax 回归的损失函数为：

$$L(\hat{y}, y) = -\sum_{i=1}^{k} y_i \log\hat{y}_i \tag{1.7}$$

这个损失函数我们又称作交叉熵损失函数。

1.5 指数加权平均

指数加权平均是一种用于求平均数的方式，它可以让某个位置的拟合值更加平滑地去平均之前位置的值。具体来说，以某一天的温度为例，设第 t 天的温度为θ_t，设这一

天的指数加权平均温度为v_t，令初始值$v_0 = 0$，指数加权平均的计算是一个递推式，具体公式如下：

$$v_t = \beta v_{t-1} + (1-\beta)\theta_t \tag{1.8}$$

展开可得：

$$v_t = (1-\beta)\theta_t + (1-\beta)\beta\theta_{t-1} + \cdots + (1-\beta)\beta^k\theta_{t-k} + \cdots \tag{1.9}$$

可以发现，由于$\beta < 1$，β^k会随着 k 的增大而减小，以天气温度为例，当天的拟合值包含对过去的平均，而越天数距今天越久远的那一天的温度就越不会对今天的温度产生影响。这样的平均就十分平滑。指数加权平均正是被应用于动量梯度下降法的计算方式。

1.6 梯度下降优化算法

假设一维参数，显然可以发现交叉熵损失函数是一个凹函数，即只存在唯一的极小值点，这个极小值点就是最小值点。对参数为 n 维的情况，由数学原理易知，n 维空间下的极小值点与在 n 维的每一维下都是极小值点是一个等价命题，因此 n 维参数的交叉熵损失函数存在唯一极小值点，且是其最小值点。而实际实验中梯度下降经常陷入了局部最优，而非全局最优。落在局部最优的根本原因实际上并不是存在多个极值点，需要注意的是一维参数下，导数为 0 的点还有驻点。当其中一些维度上是极值点，剩余维度上均为驻点，此时该点称作鞍点，此点的各方向上导数均为 0，其图线如图 1.3 所示。局部最优实际上是鞍点，因为导数为 0，相比于陷入极值点附近不再有机会降低损失，鞍点是可以继续下降的，但是在鞍点附近梯度下降会极其缓慢，让人误以为没有继续下降的空间了。n 维交叉熵函数取得极小值的前提是每一维都是极小值，取得极小值的概率等价于每一维均为极小值概率的乘积，因此取得极小值的概率远小于落在鞍点的概率。

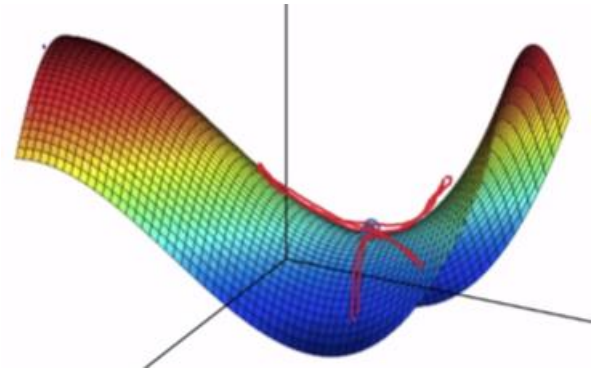

图 1.3 鞍点示意图

不使用梯度下降优化算法时，二维参数的损失函数等高线图上梯度下降示意图如图 1.4 所示，可以发现梯度下降过程中，往往会在无关方向上大幅度地来回摆动，尤其在

mini-batch 中，batch-size 越小，就越不稳定，即这种无关摆动就越明显。一方面这种摆动是计算中的浪费，另一方面增大了进入鞍部的概率，导致陷入局部最优的概率增大，因此需要梯度下降优化算法。

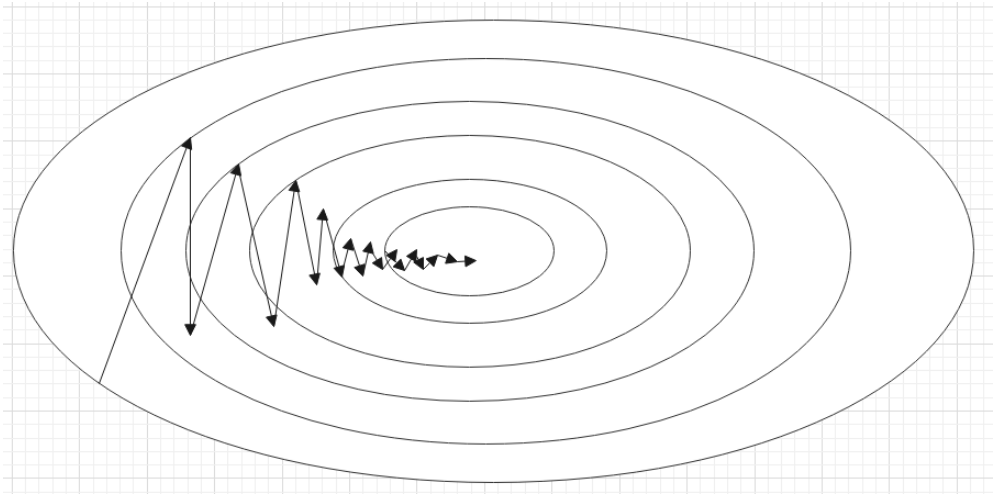

图 1.4　梯度下降示意图（四周高，中间低）

1.6.1　Momentum

动量梯度下降法，又称 Momentum。动量梯度下降法在梯度下降过程中使用了指数加权平均。设参数矩阵为 W，设已求得常规梯度下降时，W 下降的微分量dW，学习率α，本次迭代的指数加权平均为V_{dW}，上次迭代的指数加权平均为v_{dW}，则动量梯度下降法的计算公式如下所示：

$$V_{dW} = \beta v_{dW} + (1-\beta)dW \tag{1.10}$$

$$W = W - \alpha V_{dW} \tag{1.11}$$

其中β是超参数。

观察图 1.4，直观上，不妨从向量分解的角度思考，将每次的下降向量分解为平行于到最小值点方向的和垂直于到最小值点方向的，由于垂直方向上是正负来回摆动，通过指数加权平均便可以抵消掉垂直方向的摆动，水平方向的累加又使得梯度下降越来越快，因此动量梯度下降可以说是十分有效的，其中超参数β一般取 0.9 鲁棒性较好。

另外，由于初始$v_{dW} = 0$，不难发现累计的较慢，因此一般会使用偏差修正，用如下公式让前若干项的V_{dW}能够被放大：

$$V_t = \frac{V_t}{1-\beta^t} \tag{1.12}$$

其中 t 表示第 t 次迭代。由于β是一个小于 1 的数，因此随着 t 的增大，β^t会逐渐趋近 0，因此$1-\beta^t$会逐渐趋近 1，也就是后来的项便不会受到这个公式的任何影响了

1.6.2 RMSprop

RMSprop 全称 root mean square pro，均方根传播。在 Momentum 的基础上，公式做了简单的调整，如下所示：

$$S_{dW} = \beta s_{dW} + (1-\beta)(dW)^2 \tag{1.13}$$

$$W = W - \alpha \frac{dW}{\sqrt{S_{dW}+\epsilon}} \tag{1.14}$$

RMSprop 实际上是基于 AdaGrad 和 Momentum 的结合，AdaGrad 的公式差不多就是上式去掉β，这里直接对 RMSprop 进行介绍。观察图 1.4 会发现，梯度下降总是在无关方向上步幅很大，在相关方向上步幅很小，对于一个越是大的数，平方以后就越是大。我们允许摆动的存在，但只要将这样的平方历史信息累加做分母，就可以实现摆动较大时，学习步伐就小一点。但同时问题也还是，一个越是大的数，平方以后就越是大，这导致累计到历史信息中后，一直很大，导致学习步伐一直都很小，于是这就是 RMSprop 和 AdaGrad 的差别，RMSprop 结合了动量梯度下降法所用到的指数加权平均，历史信息不断乘以β，因此很快衰减，影响也就逐渐变小，解决了这一问题。另外一个问题是如果式中的$S_{dW}=0$那么学习步伐将无穷大，因此加了一个很小的数ϵ，来避免这一问题。这里ϵ是一个超参数，推荐值是10^{-8}。

1.6.3 Adam

Adam 是现在最常用的、表现最稳定的一个梯度下降优化算法，其全称是 Adaptive Moment Estimation，译为自适应矩估计。Adam 算法是对带偏差修正的 Momentum 算法和 RMSprop 算法的完完全全的一个结合，具体公式如下所示：

$$V_{dW} = \frac{V_{dW}}{1-\beta_1^t} \tag{1.15}$$

$$S_{dW} = \frac{S_{dW}}{1-\beta_2^t} \tag{1.16}$$

$$W = W - \alpha \frac{V_{dW}}{\sqrt{S_{dW}+\epsilon}} \tag{1.17}$$

其中，$\beta_1, \beta_2, \varepsilon$均为超参数，$\beta_1$称作第一矩，$\beta_2$称作第二矩，作者的推荐值为$\beta_1 = 0.9, \beta_2 = 0.999, \varepsilon = 10^{-8}$，一般情况下使用上述默认参数便可使 Adam 发挥最佳效果。

1.7 正则化

训练过程中可能会出现在验证集上效果很好，但是在测试集上效果并不好的情况，此时往往是样本预测值的方差较大。形象化地来说，样本预测值散布较大，平滑性差，

连成曲线来看就是波折比较大的曲线。主要解决方案有数据增强或正则化。正则化中比较常用的是 L2 正则化和 Dropout 正则化。

1.7.1 L2 正则化

L2 正则化的本质是在代价函数上设置了惩罚项，从而在反向传播的求导过程中，让参数以一定权重衰退。设代价函数为 J，损失函数为$L(\hat{y},y)$，样本个数为 m，参数为 w，逻辑回归中带有 L2 正则化的代价函数如下所示：

$$J(w) = \frac{1}{m}\sum_{i=1}^{m} L\left(\hat{y}^{(i)}, y^{i}\right) + \frac{\lambda}{2m}\left||w|\right|_2^2 \tag{1.18}$$

其中$\left||w|\right|_2^2$是 w 的欧几里得距离的平方，称作 L2 范数。下角标 2 用来标识它是 L2 范数，可以省略，λ是一个超参数，称作正则化参数。

上述的 w 是逻辑回归的一个参数向量 w，对神经网络而言其参数是一个矩阵，矩阵的行维度取决于该层节点的个数，设 W 为参数矩阵，设网络一共 L 层，则带有 L2 正则化的神经网络的代价函数如下：

$$J = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{i}) + \sum_{l=1}^{L}\frac{\lambda}{2m}\left||W^{[l]}|\right|_F^2 \tag{1.19}$$

其中第二范数$\left||W^{[l]}|\right|_F^2$的计算方式并没有什么差别，只是因为一些历史原因，其正式名称称作弗罗贝尼乌斯范数（Frobenius），因此用下标 F 表示，可以省略。

此时我们观察整理之后的反向传播通过求导更新 W 的公式：

$$\begin{aligned} W^{[l]} &= W^{[l]} - \alpha\left((from\ \ backward) + \frac{\lambda}{m}W^{[l]}\right) \\ &= \left(1 - \frac{\alpha\lambda}{m}\right)W^{[l]} - \alpha(from\ \ backward) \end{aligned} \tag{1.20}$$

可以看到相当于先把原来的参数 W 缩减到了$\left(1 - \frac{\alpha\lambda}{m}\right)$倍再减去学习率乘以偏导数，也因此 L2 范式属于一种权重衰退的方法。

1.7.2 dropout 正则化

dropout 正则化，又称随机失活正则化。方法是对每一层都可以设定一个 dropout 的概率，对使用了 dropout 正则化的某一层，在每一次迭代中，都会按照 dropout 的概率随机地删除这一层的一些节点，用删完节点的网络完成这一次正向传播和反向传播，对图 1.5 中的网络进行 dropout 正则化，一次迭代中 dropout 后的可能的网络如图 1.6 所示。

需要注意的是，在实际应用中我们使用的 dropout 版本实际上是优化过的 inverted dropout。需要注意如果直接 dropout，那么会导致计算结果的期望值变小。例如以 0.2 的概率进行删除，那么会导致期望值缩小为原来的 0.8 倍，因此要除以 0.8，即除以 1-dropout

概率，这就是 inverted dropout。相比于 L2 正则化的权重衰退，把权重减小的方法，dropout 正则化相当于丢掉一些权重，因此 dropout 正则化在消除过拟合方面是有效的。

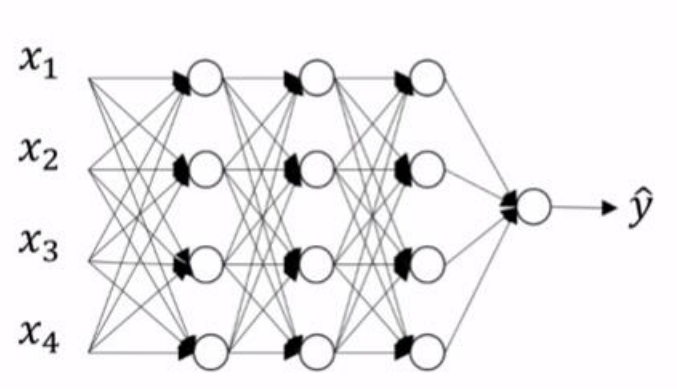

图 1.5 dropout 前的网络

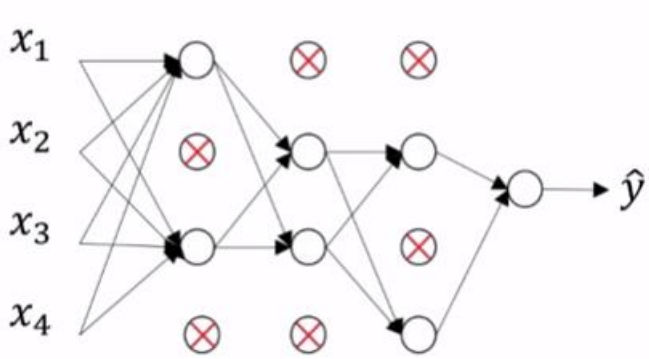

图 1.6 dropout 后的网络

1.8 注意力机制

注意力机制，作为一种资源分配方案，将有限的计算资源用来处理更重要的信息，是解决信息超载问题的主要手段。当用神经网络来处理大量的输入信息时，类比于人脑的注意力，注意力机制所实现的相当于只选择一些相关性高的信息作为输入，从而提高神经网络的效率。一种形象化的理解方式是，比如阅读理解，给定一篇很长的文章，然后给出一个问题，提出的问题可能只和句子中的某一两个句子相关，其余都是无关的。因此其余的句子没有输入的必要，也就是说只需要把相关的片段挑选出来让后续的层进行处理即可。

用$\mathrm{X}=[\mathrm{x}_1,\ldots,x_n]$来表示 N 组输入信息，只需要选择 N 组信息中一些和任务相关的信息做输入即可。注意力机制的计算可以分为两步：一是在所有输入信息上计算注意力分布或称作打分函数，二是根据注意力分布来计算输入信息的加权平均。为了选出和某个特定任务相关的信息，所以需要引入一个和任务相关的表示，称作查询向量 q。通过打分函数来计算每个输入向量和查询向量之间的相关性。权重α就是选择某个向量 x 的概率，不妨将其想象成选择某个类别的概率，这恰恰就是 softmax 回归输出的特点，于是从便于计算的角度考虑，注意力机制中一般使用的就是这样一个简易的 softmax 回归，称作“软性”的信息选择机制。选择第 n 个向量的概率权重α_n的计算公式如下所示：

$$\begin{aligned}\alpha_\mathrm{n}&=\mathrm{P}(\mathrm{z}=\mathrm{n}|\mathrm{X},\mathrm{q})\\&=\mathrm{softmax}\big(\mathrm{s}(\mathrm{x}_\mathrm{n},\mathrm{q})\big)\\&=\frac{\exp(s(x_n,q))}{\sum_{j=1}^{N}\exp(s(x_j,q))}\end{aligned}\tag{1.21}$$

其中概率权重α_n称作注意力分布，$\mathrm{s}(\mathrm{x},\mathrm{q})$称作注意力打分函数，主要有如下几种计算方式：

加性模型 $$\mathrm{s}(\mathrm{x},\mathrm{q})=\mathrm{v}^\mathrm{T}\tanh(Wx+Uq)\tag{1.22}$$

点积模型 $$\mathrm{s}(\mathrm{x},\mathrm{q})=\mathrm{x}^\mathrm{T}q\tag{1.23}$$

缩放点积模型 $$\mathrm{s}(\mathrm{x},\mathrm{q})=\frac{x^Tq}{\sqrt{D}}\tag{1.24}$$

双线性模型 $$\mathrm{s}(\mathrm{x},\mathrm{q})=\mathrm{x}^\mathrm{T}Wq\tag{1.25}$$

其中$\mathrm{W},\mathrm{U},\mathrm{v}$为可学习的参数，D 为输入向量的维度。当维度 D 很高时，点积模型会有比较大的方差，从而导致 softmax 函数的梯度较小。除以$\sqrt{D}$来放缩就可以解决这样一个问题。双线性模型的 W 可以看作线性变换$\mathrm{W}=\mathrm{U}^\mathrm{T}V$因此$\mathrm{x}^\mathrm{T}Wq=(xU)^T(Vq)$，也就是对 x 和 q 分别做了线性变换。在点积模型中，$\mathrm{x}^\mathrm{T}q$是完全对称的，并没有对二者做出本质区别，调换过来也是没问题的，是完全一样的，因此双线性模型引入了线性变换从而在计算相似度时引入了非对称性。得到了权重后，便可以对 x 加权得到聚合信息，公式如下所示：

$$\mathrm{att}(\mathrm{X},\mathrm{q})=\sum_{n=1}^{N}\alpha_n x_n\tag{1.26}$$

1.8.1 键值对注意力

更一般地，可以用键值对的形式来表示输入信息，此时用键 K 与查询向量 q 来做打分，得到的注意力分布和值 V 来加权计算聚合信息。具体公式如下所示：

$$\alpha_\mathrm{n}=\frac{\exp(s(k_n,q))}{\sum_j\exp(s(k_j,q))}\tag{1.27}$$

$$\mathrm{att}((\mathrm{K},\mathrm{V}),\mathrm{q})=\sum_{n=1}^{N}\alpha_n v_n\tag{1.28}$$

1.8.2 多头注意力

对一组查询$\mathrm{Q}=[\mathrm{q}_1,\ldots,q_M]$，可以并行地分别对每个查询向量做键值对注意力计算聚合信息，然后将结果拼接即可，公式如下所示：

$$\mathrm{att}\big((\mathrm{K},\mathrm{V}),\mathrm{Q}\big)=\mathrm{att}\big((\mathrm{K},\mathrm{V}),\mathrm{q}_1\big)\oplus\ldots\oplus\mathrm{att}\big((\mathrm{K},\mathrm{V}),\mathrm{q}_\mathrm{M}\big) \tag{1.29}$$

其中$\oplus$表示向量拼接。

1.8.3 自注意力模型

基于卷积神经网络或循环神经网络的编码都是一种局部的编码方式，尤其对于循环神经网络，存在着严重的灾难性遗忘以及梯度消失和梯度爆炸的问题，不利于长距离依赖。全连接网络可以实现长距离依赖，但是一方面无法处理边长序列，另一方面没有对权重的额外限制，自由式的学习让其不仅训练效率很差，权重的分布也难以控制得合理。自注意力机制中，对输入 X，首先分别用不同的线性变化将其映射为键值对，即$\mathrm{Q}=\mathrm{W}_\mathrm{q}X,\mathrm{K}=W_kX,V=W_VX$，然后就可以用键值对注意力来加权计算聚合信息了，一般常用的是缩放点积模型，自注意力模型的输出 H 计算公式如下：

$$\mathrm{H}=\mathrm{V}\mathrm{softmax}(\frac{K^TQ}{\sqrt{D_k}}) \tag{1.30}$$

1.9 LSTM

如图 1.7 所示，是一个一般的 RNN 结构。当然，根据任务种类的不同，RNN 的结构也有各种对应的调整，图中展示的是输入输出序列长度相等下的多对多。除此之外，比如还有一对多，多对一，和输入输出长度不等的多对多。一对多，比如序列生成，此时的结构会将上一个时间步的输出作为下一个时间步的输入。多对一，比如分类，一般就是取最后一个时间步做输出。输入输出长度不等的多对多，比如机器翻译，那这时就必须要有一个编码解码结构，使用一般的 RNN 做编码解码时，其结构一般是前半部分的时间步只有输入序列，是一个编码器，然后最后一个时间步连到解码器，只有输出序列。

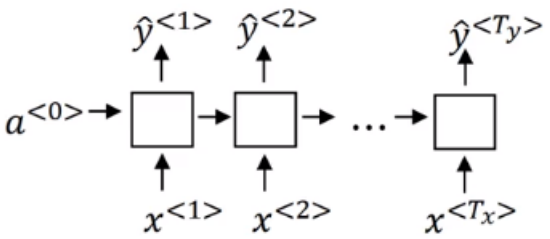

图 1.7 RNN

在一般的神经网络中，位置是无关的，因而不适合解决序列问题，RNN 的一层，通过时间步的传递，解决了这个问题[4]。然而 RNN 有着十分致命的问题，当序列较长时，训练难度将会加大，RNN 不适合长程依赖，时间步过于久远的部分很难对后面产生有效影响，存在着灾难性遗忘。而且时间步太长和一般网络结构的层数太深有着类似的问题，因为反向传播过程是一个链式求导，就是一个不断相乘的过程，如果$dW > 1$，那么大量相乘就会导致上溢到无穷，称作梯度爆炸。如果$dW < 1$，那么大量相乘就会导致下溢到 0，称作梯度消失。梯度消失和梯度爆炸导致梯度变成 nan，训练无法正常继续进行。一种强制的解决方案是梯度修剪，也就是规定一个阈值，当梯度超过这个阈值时，就将梯度变成这个最大阈值或最小阈值。但这并不能说是一个非常好的解决方法。

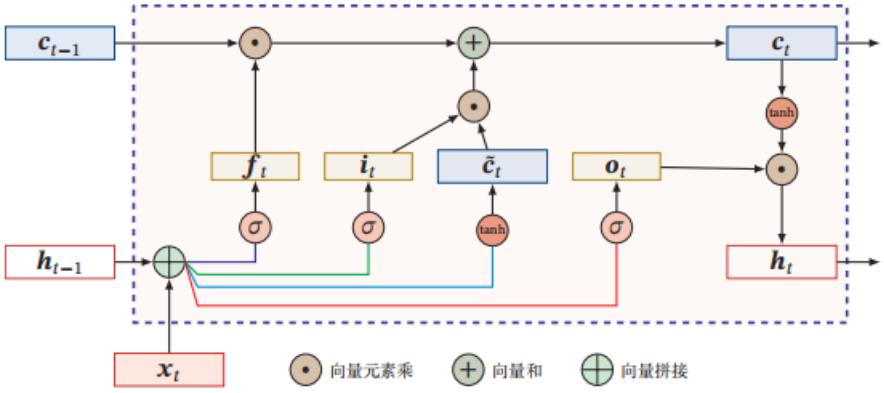

图 1.8　LSTM 循环单元

长期短期记忆网络（Long Short-Term Memory，LSTM）仿照人的记忆细胞以及数字电路中门控机制的思想，提出了一个解决方案，在过去的大量实验中证明了其确实十分有效[5]。图 1.8 所示的是一个 LSTM 循环单元，即一个时间步处的结构。

相关参数的计算公式如下：

$$\mathrm{i_t} = \sigma(W_i \cdot [h_{t-1}; w_t] + b_i) \tag{1.31}$$

$$\mathrm{f_t} = \sigma(W_f \cdot [h_{t-1}; w_t] + b_f) \tag{1.32}$$

$$\mathrm{o_t} = \sigma(W_o \cdot [h_{t-1}; w_t] + b_o) \tag{1.33}$$

$$\mathrm{c_t} = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{1.34}$$

$$\mathrm{h_t} = o_t \odot \tanh(c_t) \tag{1.35}$$

其中，$\mathrm{f_t}$、$\mathrm{i_t}$、$\mathrm{o_t}$分别称作遗忘门、输入门、输出门。内部状态$\mathrm{c_t}$，传递给外部的状态$\mathrm{h_t}$。遗忘门$\mathrm{f_t}$控制了上一时刻的内部状态c_{t-1}有多少信息需要保存。输入门$\mathrm{i_t}$控制当前时刻的候选状态$\mathrm{c_t}$有多少信息需要保存。输出门$\mathrm{o_t}$控制当前时刻的内部状态$\mathrm{c_t}$有多少信

息需要输出给外部状态h_t。不妨想象一下比较极限的状态，当$f_t = 0, i_t = 1$时，历史的记忆信息将被清空，仅将当前候选状态$\tilde{c}_t$写入c_t，这就实现了遗忘，当然需要注意的是c_t和上一时间步的历史信息还是相关的，再例如当$f_t = 1, i_t = 0$，记忆单元将复制上一时间步的内容，不写入新的信息，这就实现了不输入，因此这就是 LSTM 中的门控机制如何发挥作用的。实际应用中，实际上我们使用的是带窥视孔（peephole）连接的 LSTM，其特点是三个门不但依赖于输入x_t和上一时间步的隐藏状态h_t，也依赖于上一时间步的记忆单元c_{t-1}。

1.10 Transformer

Transformer 可以说是近几年研究中最具影响力的模型之一，Transformer 的架构中仅使用了注意力机制，开创了继 MLP、CNN、RNN 以来的第四大类模型[15]。Transformer 的整体架构如图 1.9 所示。

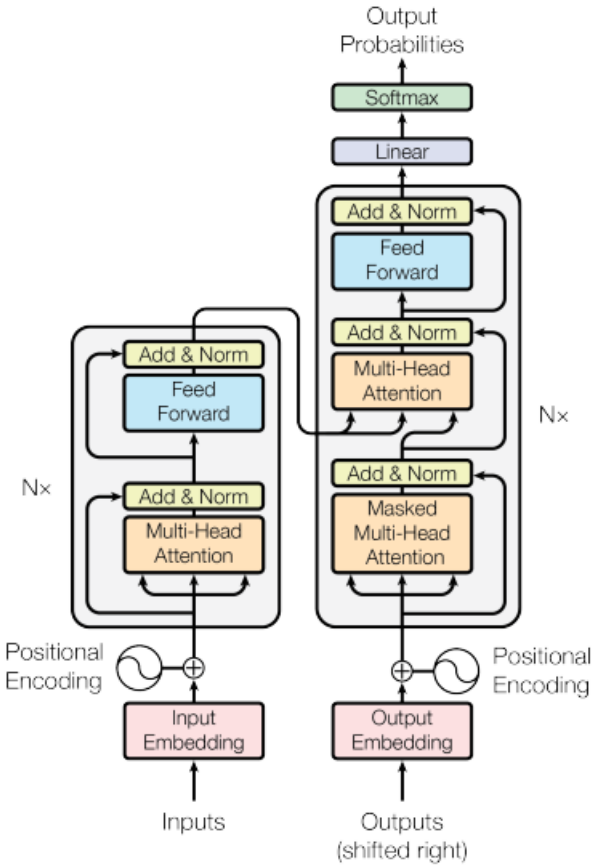

图 1.9 Transformer 模型架构[15]

Transformer 是一个编码器-解码器结构，由 N 层编码器和 N 层解码器构成，作者推荐的大小是 N=6。在一层编码器中，先是一个对输入的多头自注意力，然后做了层归一化（layer norm），注意其中对输入有一个残差连接到归一化，然后连到一个前馈层，实际上就是 MLP，然后再层归一化。过程中的维数为了方便，作者将词嵌入的维数和序列长均填充到了 512。这里多头自注意力用了 8 个，也就是对键值分成长度 64 的八块分别计算后拼接。对 N 层解码器的输出作为 K 和 V 输入到了解码器的多头自注意力子层。这里注意解码器唯一的区别是加了一个带掩码的多头自注意力，也就是将当前位置以后的通过乘以掩码矩阵变成 0，这样实现了当前位置的输出不会受到后面位置的影响，仅看前面的位置。

当然，此时的 Transformer 有一个很明显的问题就是，实际上他与序列顺序是完全无关的，即便颠倒位置，也不会改变自注意力的结果，于是这里就加入了一点特征工程方法，对嵌入后的矩阵加了一点位置编码（Positional Encoding，PE），公式如下：

$$\mathrm{PE}(\mathrm{pos}, 2\mathrm{i}) = \sin(\mathrm{pos}/10000^{2\mathrm{i}/\mathrm{d}_{model}}) \tag{1.36}$$

$$\mathrm{PE}(\mathrm{pos}, 2\mathrm{i}+1) = \cos(\mathrm{pos}/10000^{2\mathrm{i}/\mathrm{d}_{model}}) \tag{1.37}$$

Transformer 的最大好处是没有序列的传递，这让它并行性十分好，效率非常高。然而对位置信息的捕捉较差是一个比较大的缺点。键值对的注意力机制让 Transformer 在 seq2seq 任务中表现得非常好，原作者在机器翻译任务中使用 Transformer 取得了当时最先进的表现。

1.11 BERT

BERT 全称 Bidirectional Encoder Representation from Transformers，可以译作 Transformers 的双向编码器表示，BERT 可以说是近几年自然语言处理最重要的文章，开创了一个全面预训练的时代[18]。BERT 的整体架构如图 1.10 所示。

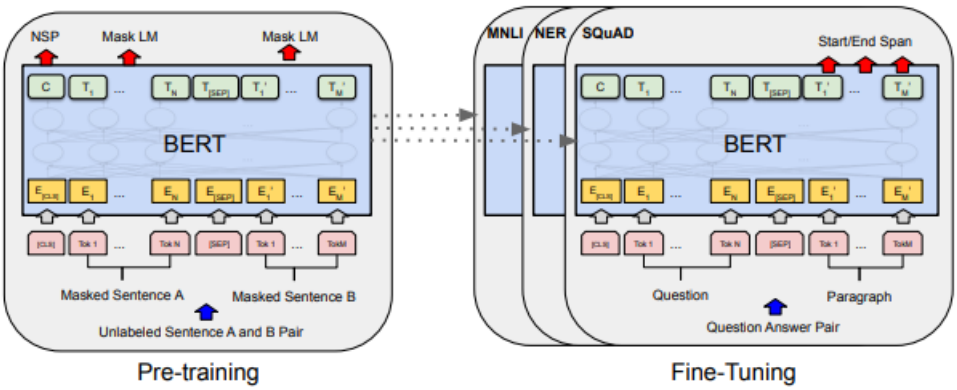

图 1.10 BERT 模型架构[18]

BERT 使用了 Transformer 的 Encoder，在大型语料库上通过无监督学习来训练词嵌入。对每个输入，第一个 token 永远是[CLS]，然后对于句子对，之间用[SEP]这个 token 进行分隔。输入到 BERT 中的是 token 的嵌入和 Segment 的嵌入和位置的嵌入之和，这里 Segment 指的是哪一个句子，也就是[SEP]分隔的句子中的哪一个。具体预训练的方式类似于完形填空。任务一是 Masked LM，一个词 15%的概率作为需要被预测的标签，对于这个词，80%概率换成[MASK]、10%概率换成随机 token，10%概率不变，之所以不是 100%换成[MASK]是因为微调任务中[MASK]不可能出现，因此 100%换成[MASK]是没有意义的。任务二是，50%的概率来选择 B 句子是否是 A 的下一句，预测的标签是 B 是否是 A 的下一句。BERT 的效果效果非常的好，过去的研究证明 BERT 在自然语言处理领域几乎所有任务中都取得了最先进的效果，由于使用了大型语料预训练，对小样本微调任务的提升尤为明显。

1.12 Flask 框架

Flask 框架是 python 中一个简单灵活、轻量级、可定制的 WEB 开发框架，保持代码简洁的同时，又有着丰富的可扩展性和强大的兼容性，也因此被称作微框架。Flask 的主要特点如下所示：

（1） Flask 主要包括 Werkzeug 和 Jinja2 两个核心函数库，他们分别负责业务处理和安全方面的功能，这些基础函数为 Web 项目开发过程提供了丰富的基础组件。

（2） Flask 中的 Jinja2 模板引擎，提高了前端代码的复用率。可以大大提高开发效率并且有利于后期的开发与维护。

（3） Flask 不会指定数据库和模板引擎等对象，用户可以根据需要自己选择各种数据库。

（4） Flask 不提供表单验证功能，在项目实施过程中可以自由配置，从而为应用程序开发提供数据库抽象层基础组件，支持进行表单数据合法性验证、文件上传处理、用户身份认证和数据库集成等功能。

1.13 本章小结

本章详细介绍了本文需要使用到的语言、类库。介绍了 python 语言的特点以及 numpy、pytorch 的优点，也介绍了当前在自然语言处理领域尤为流行的 Transformers 预训练库。然后详细介绍了本文所涉及的数学原理，以及一些基本模型。

2 系统方法

2.1 TD-LSTM

TD-LSTM[7]方法的架构如图 2.1 所示。

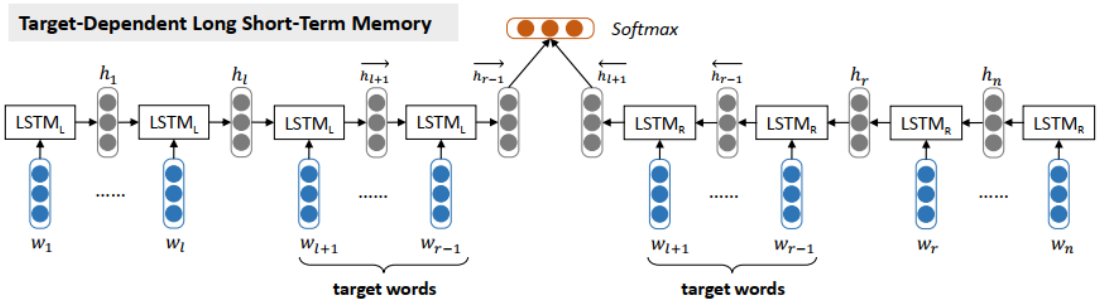

图 2.1 TD-LSTM 方法架构[7]

TD-LSTM[7]在引言中已有一定的介绍，具体来说将方面词左边的上文和方面词拼接，嵌入后输入到左边的一个 LSTM-L 中。将方面词和方面词右边的下文拼接，嵌入后输入到右边的 LSTM-R 中，LSTM-R 按逆向的序列顺序进行时间步的传播，或者，实际上实际操作中通过相对的思想对序列进行翻转即可。对左右 LSTM 取方面词最后一个位置的外部状态 h 的输出，将二者拼接后输入到 softmax 层输出分类。其中一定需要注意的一个细节是 LSTM 的输出是方面词的最后一个位置，而非填充序列的最后一个位置，即输出的位置是当前序列最后一个非 0 的位置，因此减少了无意义的时间步传播，避免了不必要的遗忘，因此十分有效。

2.2 TD-Transformer

Ashish Vaswani 等人在提出 Transformer 时，将其视作对 CNN、RNN 的一种取代，表示 CNN、RNN 并不是必要的[15]。因此本文尝试使用 Transformer 来替代 LSTM，并提出了目标依赖的 Transformer（Target-Dependent Transformer，TD-Transformer）。TD-Transformer 架构如图 2.2 所示。

经过尝试，发现由于数据集较小，并不适合较深的网络，因此最终 TD-Transformer 仅使用了一层 Transformer Encoder。为了尽可能利用位置信息，因此使用了掩码多头自注意力并且对输入加入了位置编码，对二者的介绍已在 1.11 节中说明。PE 的公式见公式 1.37 和公式 1.38。其中输入的嵌入维数按 Transformer 作者的推荐使用的是 512，这样也便于保持使用推荐的 8 头的多头自注意力[15]。而为了方便以及便于比较，因此加入

位置编码的操作是对 glove-300d 得到的 300 维嵌入，拼接了一个广播的 212 维位置编码。序列最长长度保持为 85，具体说明见 3.2 节。对 Encoder-L 和 Encoder-R 分别取方面词位置的输出拼接输入到 softmax 层，和 TD-LSTM 是保持一致的，这也体现了使用掩码的意义。

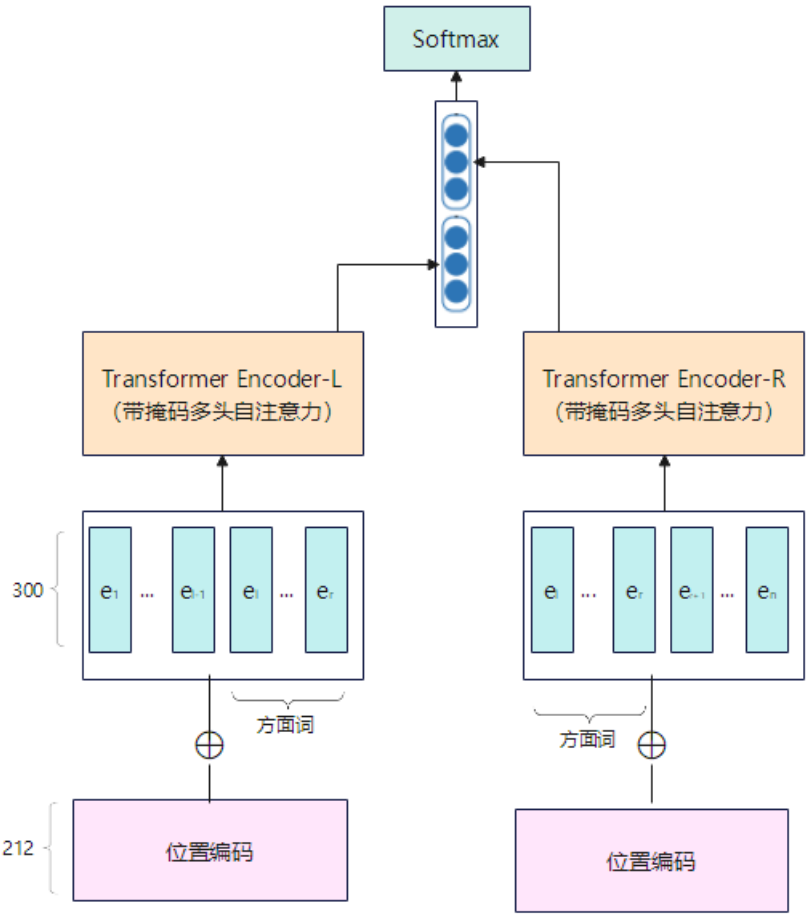

图 2.2　TD-Transformer 方法架构

2.3　TDLC-Transformer

在 3.3 节中，可以看到 TD-Transformer 的表现还远远无法令人满意。考虑到 LSTM 依然会受到长程依赖的制约，以及 Transformer 难以对位置信息加以利用，因此对二者尝试进行了结合。在 1.8.3 节中给出了对自注意力模型的介绍，借用 Cache 和主存映射的一种描述方法，从中不难发现，自注意力模型和全连接模型都相当于一种从输入 X 到状态 H 的带权全相联映射。不妨思考其中的差别，自注意力模型的权重实际上是在打分

函数的限制下动态学习生成的，而全连接模型则是在无任何制约的情况下完全依赖于反向传播学习得到权重。一种形象化的解释是，自注意力模型会根据相似度对映射的选择加以限制，形象化的图线表示如图 2.3、图 2.4 所示，其中虚线表示动态生成。

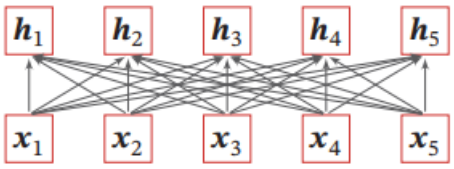

图 2.3　全连接模型映射表示

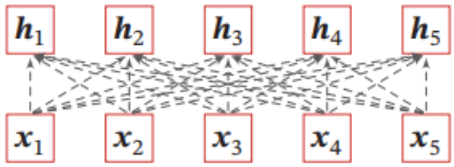

图 2.4　自注意力模型映射表示

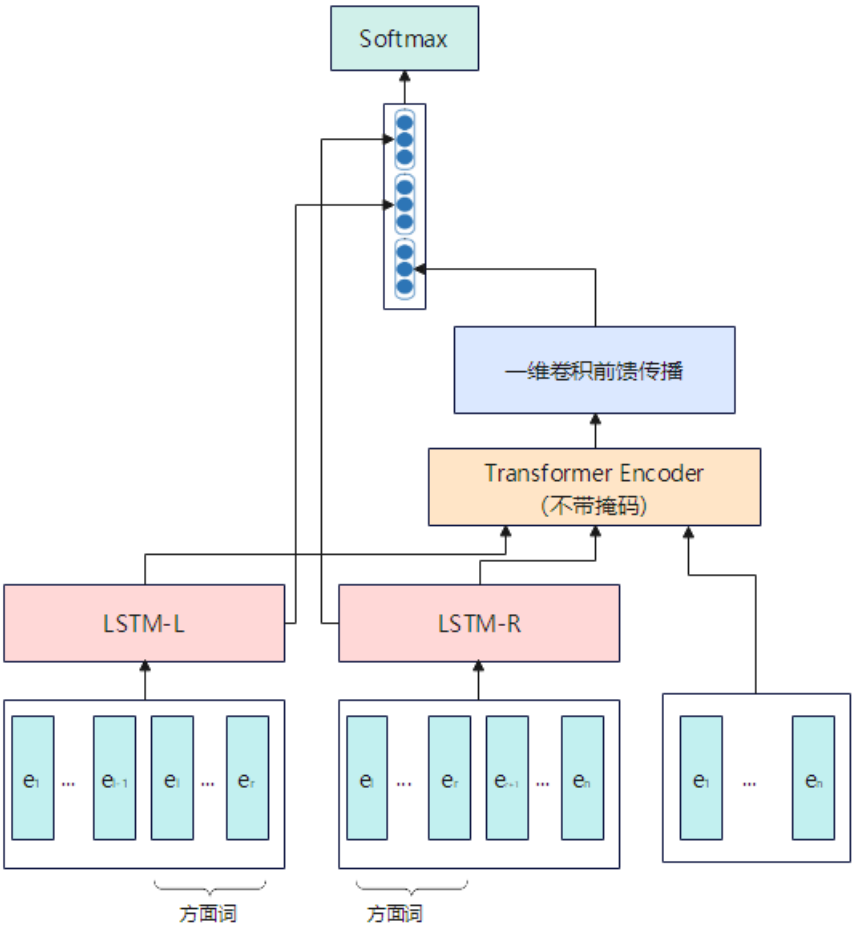

图 2.5　TDLC-Transformer 方法架构

因此从这样的角度理解，因此本文提出让自注意力在 TD-LSTM 的 LSTM-L、LSTM-R 以及原始序列的嵌入矩阵 X 这三者中进行自我选择。由于 Transformer 的 Encoder 中

的多头注意力、残差连接和归一化都是很有必要的，因此为了方便并没有使用单纯的自注意力，而是直接使用了 Transformer Encoder，提出了目标依赖的 LSTM 连接的 Transformer（Target-Dependent LSTM Concatenate Transformer，TDLC-Transformer），其架构如图 2.5 所示。

具体来说我将 LSTM-L、LSTM-R 的输出和原始嵌入矩阵 X，这三者在序列维度上拼接，输入到一层 Transformer Encoder 中，由于对 LSTM 进行选择，再使用 PE 和 Mask 的意义不是很大，因此 PE 和 Mask 均未使用。同时也是为了方便，Transformer Encoder 的嵌入维数设置在了 300，由于需要整除的原因，多头自注意力设置为了 6 头。对 Transformer Encoder 的整个序列，按嵌入位置，将序列中所有词，全连接映射为一个词嵌入向量。具体来说，就是序列长做通道维，对维数 300 的嵌入维做卷积核大小为 1 的一维卷积，输出通道数设为 1，从而得到一个嵌入维数 300 的向量，然后我将这个 300 维向量与 LSTM-L、LSTM-R 方面词位置输出的 300 维向量进行拼接，得到一个 900 维向量输入到 softmax 输出层进行分类。

2.4 TAGG-LSTM

在 3.3 节中可以看到，TDLC-Transformer 的表现只在 Twitter 数据集上有了明显提高，在其余两个数据集上有非常小的下降。因此本文进一步考虑抛开 Transformer，对 LSTM 方法架构进一步调整是否能提高在所有数据集上的表现。Tang 等人的实验表明了 TD-LSTM 在训练集上大量训练后仅在 Twitter 数据集上有着较大的欠拟合[6]，最后表现的主要制约是网络结构和 LSTM 本身的潜能上限导致的泛化能力不足。尽管过去的实验中，并未表现出严重的过拟合，但进一步消除过拟合确实有可能缩短测试集与训练集上表现的差距。基于泛化能力和尽可能多消除一些过拟合的双重考虑，本文提出了一种基于目标聚合的 dropout 正则化方式（Target-Aggregation dropout，TAGG-dropout），以及以该 dropout 方式为基础的基于目标聚合的 LSTM 方法（Target-Aggregation LSTM，TAGG-LSTM）。非常可惜的是，在 3.3 节中可以看到 TAGG-LSTM 的表现结果有着和 TDLC-Transformer 一样的弊病，但可能是 TAGG-LSTM 层数较深训练难度较大导致的。

2.4.1 TAGG-dropout

TAGG-dropout 的提出主要是基于多层 RNN 结构的，其前提是任务依赖于目标。TAGG-dropout 的失活方式是，每层按照固定频率*输入序列长，从远离目标端进行固定数据的失活。例如图 2.6 所示，图中 dropout 频率为 0.2，层数为 3 层，输入序列长为 5，因此每层的输出中从远离目标端依次失活 0.2*5=1 个 RNN 单元后输入进下一层。

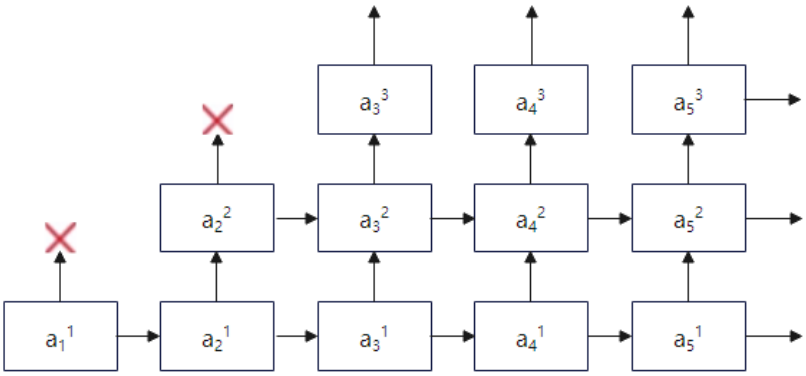

图 2.6　TAGG-dropout

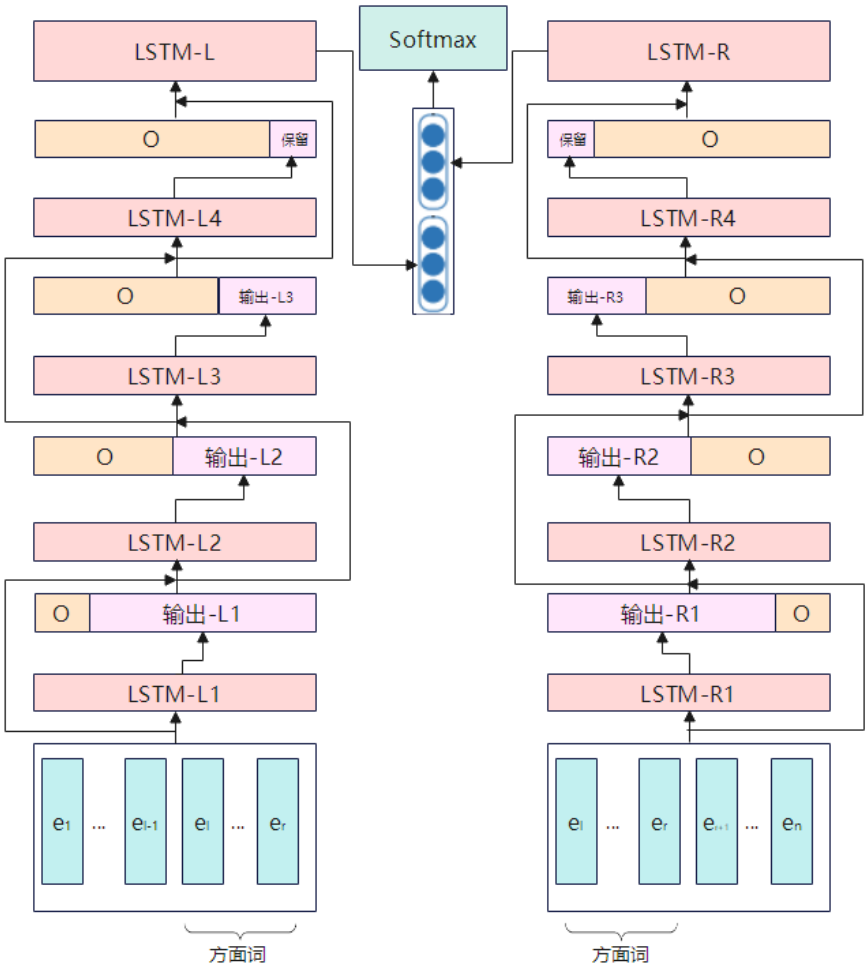

图 2.7　TAGG-LSTM 架构

2.4.2 TAGG-MASK

具体操作中，使 RNN 单元失活采用的是掩码方式，创建全 1 的掩码矩阵 M，在需要失活的序列前部的对应位置上为 M 赋为 0，和候选状态对应点位相乘即可。即对序列长 len，失活频率 p，TAGG-dropout 的第 l 层（$l \geq 1$）候选状态$\widetilde{H_l}$，计算器输出状态H_l如下：

$$m_{ij} = \begin{cases} 0 & j < len_i * l * p \\ 1 & j \geq len_i * l * p \end{cases} \tag{2.1}$$

$$M = [m_0^T, m_1^T, \ldots, m_n^T]^T \tag{2.2}$$

$$H_l = \widetilde{H_l} \circ M \tag{2.3}$$

其中○表示对矩阵对应位置相乘的计算。

2.4.3 TAGG-LSTM 架构

由 LCF-BERT 的带来的灵感，将语境聚集到方面词的一定范围内可以带来更好的效果。在 3.4 节中表 3.5 给出了 LCF-BERT 作者对语义相对距离 SRD 的推荐，某词的语义相对距离即某词到方面词最左端和最右端中的最短距离，某词的语义相对距离小于等于设置的 SRD 被认为是局部语境。对表 3.5 的 SRD 进行平均，可得到推荐的 SRD 平均值约 4.5。再由表 3.3 得方面词的平均长度约为 1.5。因此对于上/下文+方面词的一句话，将其聚合到长度为 4.5+1.5=6 是比较合适的。常用的 dropout 概率一般是 0.1 或者 0.2。在表 3.3 中可以看到有一些特例的句子非常的长。另外，层数过深也不利于小样本数据集的训练。同时考虑这三点，因此 TAGG-dropout 的频率不宜过小，也不宜过大，因此本文将 TAGG-dropout 的频率设置为了 p=0.25，通过对距方面词最后一个位置 6 以外的词进行四层聚合，即 l=4，来使序列聚合到长度 6，另外，为了保证模型的稳定性，在每一层的输出上都加上了残差连接。具体操作的公式如下：

$$m_{ij} = \begin{cases} 0 & j < (len_i - 6) * l * p \\ 1 & j \geq (len_i - 6) * l * p \end{cases} \tag{2.4}$$

$$M = [m_0^T, m_1^T, \ldots, m_n^T]^T \tag{2.5}$$

$$H_l = H_{l-1} + \widetilde{H_l} \circ M \tag{2.6}$$

其中符号设定同 2.4.2 节。通过残差连接的不断相加，由于被加的前部是不断掩码的，实际相加的是后部，因此其另一个好处就是进一步增强了对目标信息的聚合。最后我对聚合后的结果又加上了一层 LSTM 进行汇总，之后的处理就和 TD-LSTM 是一样的。TAGG-LSTM 方法架构如图 2.7 所示。

2.5 LCF-BERT

在 ABSA 这类小样本学习的任务中，不使用预训练模型和使用预训练模型之间，在表现上有着十分明显的巨大差距。事实表明，使用一般的 Transformer 远远不如大型语料库预训练好的 BERT。考虑到本系统最终要部署，因此准确率是很重要的，因此对已被证实有效的 LCF-BERT 进行尝试。同时也对领域语料继续预训练能否进一步提高表现进行了进一步的尝试，并对能否提高的相关因素进行了进一步研究。在引言中已对 LCF-BERT 进行简要介绍，这里在详细地说明一下每部分结构，LCF-BERT 架构如图 2.8 所示[21]。

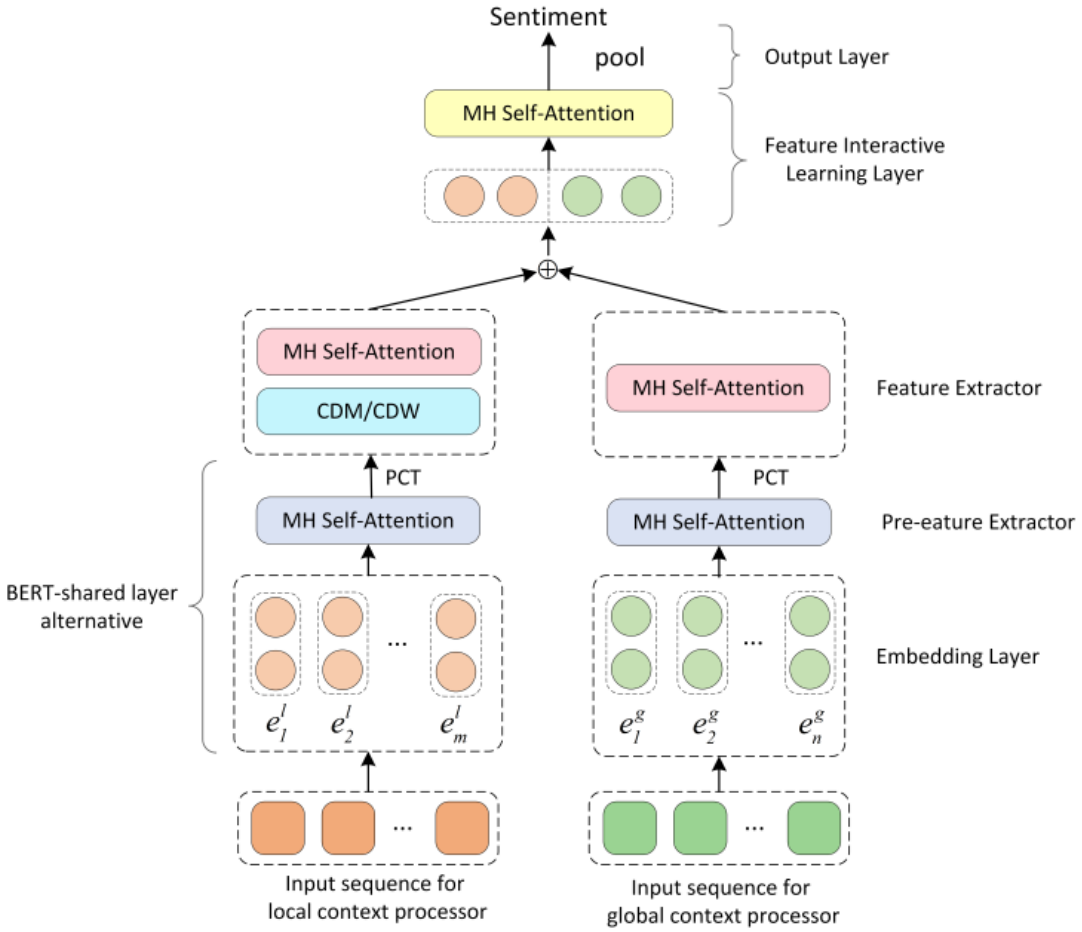

图 2.8 LCF-BERT 架构[21]

对输入序列做两份处理，一份做局部语境处理，另一份做全局语境处理。首先这两份输入序列都使用 BERT 先做词嵌入（可以用 glove+多头自注意力 MHSA 替代，速度

快很多，但效果也差一些），对局部语境嵌入后首先按照超参数 SRD，根据选择做 CDM 或 CDW，然后做一个多头自注意力。对全局语境嵌入后直接做一个多头自注意力。将局部和全局通过多头自注意力后的结果直接拼接，再做一个多头自注意力，然后通过 bert_pooler，从[CLS]获得池化，最后接入 softmax 层得到分类结果。

2.5.1 语义相对距离

语义相对距离（Semantic-Relative Distance，SRD）。通俗的理解就是某个词到方面词最左端或最右端的最小距离，设方面词长度为 m，方面词的中心位置为P_a，位置 i 处的词的语义相对距离SRD_i的公式如下：

$$SRD_i = |i - P_a| - \left\lfloor \frac{m}{2} \right\rfloor \tag{2.7}$$

SRD 是一个超参数，其推荐值在 3.2 节的表 3.5 中。局部语境需要聚焦到 SRD 范围内，对$SRD_i > SRD$范围内的词需要做局部语境聚焦处理，具体操作如下一小节中所述。

2.5.2 局部语境聚焦

局部语境聚焦（Local Context Focus，LCF），具体实现方式有两种，分别是语境特征动态掩码（Context features Dynamic Mask，CDM）和语境特征动态加权（Context features Dynamic Weighted，CDW）。对于 CDM，就是不在 SRD 范围内的输出值直接设为 0，即消除掉了输入对不在 SRD 范围内输出的权重。对于 CDW，就是不在 SRD 范围内的输出值按到方面词的距离反比做权重衰减，即输入对不在 SRD 范围内输出的权重衰减。设 BERT 嵌入层输出O^l_{PFE}，CDM 输出O^l_{CDM}，CDW 输出O^l_{CDW}，设 SRD 超参数值为α。

CDM 计算公式如下：

$$V_i = \begin{cases} E & SRD_i \le \alpha \\ O & SRD_i > \alpha \end{cases} \tag{2.8}$$

$$M = [V^m_0, V^m_1, \dots, V^m_n] \tag{2.9}$$

$$O^l_{CDM} = O^l_{PFE} \cdot M \tag{2.10}$$

CDW 计算公式如下：

$$V_i = \begin{cases} E & SRD_i \le \alpha \\ \frac{SRD_i - \alpha}{n} \cdot E & SRD_i > \alpha \end{cases} \tag{2.11}$$

$$M = [V^w_0, V^w_1, \dots, V^w_n] \tag{2.12}$$

$$O^l_{CDM} = O^l_{PFE} \cdot M \tag{2.13}$$

2.6 本章小结

本章对主要依赖的过去的方法 TD-LSTM 和 LCF-BERT 的架构和一些计算公式进行了详细的描述，对其原理本质进行了简单的剖析。并对本文提出的三个新的方法，TD-

Transformer、TDLC-Tranformer、TAGG-LSTM 的架构进行了详细描述并阐明了研究过程中的一些思路。

3 系统实验

3.1 数据集

数据集使用的是过去十年最常用于比较的三组数据集，分别是 SemEval 在 2014 年提出该任务时发布的 Resturants-14 餐馆评论数据集和 Laptops-14 笔记本评论数据集，以及 ACL 在 2014 发布的 Twitter 推特评论数据集。数据集中，特征是一个英文句子和其中的方面词，标签是其对应的情感极性。情感极性的取值范围是$[-1,0,1]$这三个数，其中 -1 代表消极，0 代表中性，1 代表积极。训练集和测试集的长度如表 3.1 所示。

表 3.1 数据集长度

数据集名称	训练集	测试集	合计
Laptops-14	2328	638	2966
Restaurants-14	3608	1120	4728
Twitter	6248	692	6940

可以观察到 laptops-14 数据集偏短，Twitter 数据集偏长。其中 Twitter 数据集又训练集占比较大。

不同情感极性的数据条数如表 3.2 所示。

表 3.2 不同情感极性的数据条数

数据集名称	数据集部分	积极	中性	消极
Laptops-14	训练集	994	464	870
	测试集	341	169	128
	合计	1335	633	998
Restaurants-14	训练集	2164	637	807
	测试集	728	196	196
	合计	2892	833	1003
Twitter	训练集	1561	3127	1560
	测试集	173	346	173
	合计	1734	3473	1733

可以观察到，其中 Restaurants-14 数据集和 Laptops-14 数据集含有中性情感极性的条数明显较少，而 Twitter 数据集含有的中性情感极性的数据条数明显较多，达到了数据集的约 50%。

实验中还对方面词和左右上下文的长度进行了进一步统计分析，如表 3.3 所示。

表 3.3　数据集的句词长度

数据集名称	上文长度			下文长度			方面词长度			整句长度	
	最小	最大	平均	最小	最大	平均	最小	最大	平均	最小	最大
Laptops-14	1	70	10.1	1	75	9.7	1	7	1.51	3	83
Restaurants-14	1	70	9.1	1	71	9.3	1	19	1.40	3	79
Twitter	1	37	7.8	1	38	11.3	1	3	1.70	3	45
所有数据集	1	70	8.7	1	75	10.3	1	19	1.57	3	83

在预训练过程中，使用了领域语料做了进一步预训练，为了研究与话题是否有关，数据集按照话题进行划分为 Restaurants 餐馆、Laptops 笔记本和 Twitter 推特。对于餐馆和笔记本数据集，SemEval 在 2014 年到 2016 年连续提供了三份数据集，而其中 2016 年的数据集是对 2015 年数据集的修正和扩展，因此领域内数据集使用了 2014 年和 2016 年的所有训练集和测试集的合并。Twitter 数据集由于没有额外的，使用的还是其本身训练集和测试集的合并。All 数据集是三个预训练数据集的合并。预训练中使用的数据集长度如表 3.4 所示。

表 3.4　继续预训练使用的领域内数据集长度

数据集名称	长度
Laptops	6274
Restaurants	7404
Twitter	6940
All	20618

另外，关于任务中训练集的划分与使用。超参数调整过程中将训练集分成 80%训练集和 20%验证集，超参数的设置以各超参数原作者推荐值为基准，以对验证集预测的表现为参考，在一定程度上进行小幅调整。在测试过程中，使用完整的训练集进行训练。

3.2 超参数设置

梯度下降优化器使用 Adam，Adam 的超参数使用作者的推荐值。梯度下降过程中权重衰减使用 L2 正则化，除 LCF-BERT 以外的模型 L2 正则化参数设为 0.01。LCF-BERT 按作者的建议 L2 正则化参数设置为 1e-5。从方差一致性的好处出发，因此网络参数的初始化使用的是 Xavier 均匀分布。batch_size 统一设置为 16。由表 3.3 的最长长度，将序列长设置为 85，不足需要 padding 填充。词嵌入的维数，除 LCF-BERT 以外的模型，由于使用 Glove-300d，因此词嵌入维数 300，LCF-BERT 用 BERT-BASE 版本，词向量维数按推荐的 768。除 LCF-BERT 以外的方法学习率使用 1e-3，LCF-BERT 由于 BERT 需要一个非常小的学习率来收敛，大的学习率反而可能发散，因此设为 Sun 等人推荐的 2e-5[25]。所有方法均未使用 dropout，特别地，TAGG-LSTM 中，可以视作人工 dropout，dropout 概率为 0.25。在 LCF-BERT 中，对局部语境的 CDM、CDW 方法均进行了尝试，超参数 SRD 按提出者的推荐对不同数据集以及不同局部语境聚焦方法进行不同设置[21]，具体如表 3.5 所示。

表 3.5 不同数据集及局部语境聚焦方法下的 SRD 设置[21]

局部语境处理方法	Laptops-14	Restaurants-14	Twitter
CDM	6	3	3
CDW	3	7	5

对于继续预训练 BERT，最大序列长设置为 768，batch_size 设置为 4，在单个话题的预训练数据集上预训练 6 个 epochs（约 10k steps），在合并的 All 数据集上预训练 2 个 epochs（约 10k steps）。使用任务一 MLM（Masked Language Model）继续预训练，掩码方法完全按照原作者的推荐，80%概率替换为[MASK]，10%概率替换为随机 token，10%概率不变[18]。

另外实验中 random、numpy、torch 所有的随机种子均固定设置为 1234，具有完全的可复现性。

3.3 方法比较

在早期的一些方法中，实验还不是很全面，部分数据集没有做，而且 TD-LSTM[6]和 ATAE-LSTM[7]中使用的词嵌入仍是 glove-200d。另外 TD-LSTM 的作者曾通过实验表示在 TD-LSTM 的实验中 glove-200d 相比于 glove-100d 没有明显提升，而训练时间明显延长。一方面为了方便与其他方法进行比较，另一方面为了研究 glove-300d 相较于 glove-

200d 对结果影响的差异，本实验中对 TD-LSTM 和 ATAE-LSTM 使用了 glove-300d 重新进行了实验。

表 3.6　不同方法实验结果表现的比较

	方法	Laptops-14		Restaurants-14		Twitter	
		准确率 (%)	F1 分数 (%)	准确率 (%)	F1 分数 (%)	准确率 (%)	F1 分数 (%)
过去的实验	Feature-based SVM	-	-	-	-	63.40	67.30
	RecNN	-	-	-	-	66.30	65.90
	$\text{TD-LSTM}_{\text{glove-200d}}$	68.13	-	75.63	-	70.8	69
	$\text{ATAE-LSTM}_{\text{glove-200d}}$	68.7	-	77.2	-	-	-
	$\text{MemNet}_{\text{glove-300d}}$	70.33	64.09	78.16	65.83	68.50	66.91
	$\text{RAM}_{\text{glove-300d}}$	74.49	71.35	80.23	70.80	69.36	67.30
	$\text{BERT}_{\text{BASE}}\text{-SPC}$	78.99	75.03	84.46	76.98	73.55	72.14
	$\text{AEN-BERT}_{\text{BASE}}$	79.93	76.31	83.12	73.76	74.71	73.13
	$\text{TD-BERT}_{\text{BASE}}$	78.87	74.38	85.10	78.35	76.69	74.28
	$\text{LCF-BRET}_{\text{LARGE}}\text{-CDM}$	82.29	79.28	86.52	80.4	76.45	75.52
	$\text{LCF-BRET}_{\text{LARGE}}\text{-CDW}$	82.45	79.59	87.14	81.74	77.31	75.78
本文的实验	$\text{TD-LSTM}_{\text{glove-300d}}$	69.28	62.37	76.88	61.79	70.09	66.67
	$\text{ATAE-LSTM}_{\text{glove-300d}}$	71.16	66.05	78.30	65.35	68.06	66.09
	$\text{TD-Transformer}_{\text{glove-300d}}$	66.14	59.53	74.11	61.04	70.81	68.55
	$\text{TDLC-Transformer}_{\text{glove-300d}}$	68.97	62.18	76.43	63.92	71.82	69.66
	$\text{TAGG-LSTM}_{\text{glove-300d}}$	67.87	60.96	76.70	64.45	71.53	69.35
	$\text{LCF-BRET}_{\text{BASE}}\text{-CDM}$	79.62	76.01	84.11	76.44	73.70	72.37
	$\text{LCF-BRET}_{\text{BASE}}\text{-CDW}$	79.47	75.75	85.09	77.64	74.42	73.72

对于本文提出的 TD-Transformer、TDLC-Transformer、TAGG-Transformer 三个方法，也均统一使用了 glove-300，便于比较。LCF-BERT 的作者在文中使用的是 BERT 的 LARGE 版本，而且比较的对象仅是对 BERT-PT、BERT-SPC 这两个 BERT 基线模型使

用 LARGE 版做的复现[21]。在过去的实验中，绝大多数作者选用的都是 BASE 版本，这为比较带来了不便。因此本文中对 LCF-BERT 使用 BASE 版的 BERT 进行了重新实验。另外，使用 BASE 版 BERT 的原因还有两方面，一方面对 BASE 版 BERT 和 LARGE 版 BERT 对结果的影响再一次进行了确认，另一方面 LARGE 版 BERT 和 BASE 版 BERT 参数量差距约为三倍[18]，考虑到最后本系统要进行部署，LARGE 版的运行效率难以接受，因此本系统全程使用的都是 BASE 版的 BERT，另外使用 BASE 版 BERT 训练也快很多，节约了大量的研究时间。过去实验中最常被用于比较的方法以及本实验中使用的方法的实验结果的表现如表 3.6 所示，其中使用预训练的 glove 词嵌入模型的方法对 glove-200d 或 glove-300d 进行了标注，使用 BERT 的方法对使用 BERT 的版本是 BASE 还是 LARGE 进行了标注。

从表 3.6 中可以看到，整体来看，使用了 glove-300d 后，TD-LSTM 和 ATAE-LSTM 的表现有了较为明显的提升，准确率整体提升达到了约 1%。但其中，在 twitter 数据集上 TD-LSTM 使用了 glove-300d 后相比 200d 有了表现轻微下滑，原因可能是 twitter 数据集可能更加看重句子结构，对词义的扩充可能反而带来了一定噪音。接下来比较 300d 下的 TD-Transformer 和 TD-LSTM，整体来看 TD-Transformer 的表现和 TD-LSTM 相比有着明显的差距，主要集中在 Laptops-14 和 Restaurants-14 数据集上，而在 twitter 数据集上 TD-Transformer 反而有了轻微的提升。原生的 Transformer 由于注意力机制的制约，显然还是不是很适合序列性较强的问题。其中 twitter 数据集中评论用语相对随意，可能序列性并不是很强，可能方面词对局部语境的依赖可能没有那么强，长程依赖会多一些，因此完全不受灾难性遗忘的 Transformer 模型表现更优。然而 ATAE-LSTM 同样使用注意力机制，却在 twitter 数据集上表现明显差很多，考虑到 ATAE-LSTM 将方面词视作查询向量，而 TD-Transformer 是整句的自注意力，所以原因还是推特数据集可能方面词的决定性作用偏弱。接下来对于 TDLC-Transformer，将 LSTM 和 Transformer 结合了之后，确实起到了明显的优势互补作用，主要是在 Twitter 数据集上，表现得明显要好很多。然而在 Laptops-14 和 Restaurants-14 数据集上，相比 TD-LSTM 变化不是很大，而且还是有小幅的下滑。在这两个数据集上，可能可以说是 Transformer 拉低了 TDLC-Transformer 的表现，Transformer 并不适合于解决这两个数据集，所幸被 LSTM 抬高，比 TD-Transformer 还是好很多的。TAGG-LSTM 的在 Laptops-14 上的表现还要更低一些，但是在 Twitter 数据集上有了非常明显的提高，TAGG-LSTM 可以更好的对方面词进行汇聚，而现在这个结果似乎与前面的分析矛盾，看起来 twitter 才是过度依赖方面词的那个数据集。实际上我们不妨把这个结果和表 3.1 中的数据集长度进行对应，不难发现明显数据集越长效果就越好。实际上 TAGG-LSTM 的结构是非常有效的，然而对于小

样本学习而言层数过多，过小的数据集在这个偏大一点的网络上拟合难度很大，这也就导致了表中的这样一个结果。对于 LCF-BERT，显然 CDW 结构还是比 CDM 结构要好一点。BERT 的 BASE 和 LARGE 版的差距，基本就和 BERT 作者实验的结果差不多，差距约 2%到 3%[18]。通过对 BASE 版 BERT 的实验，实际上可以发现 TD-BERT 总体上比 LCF-BERT 要好一点点，而二者都明显要比 AEN-BERT 好一点。

3.4 领域语料继续预训练 BERT

表 3.7 话题内和跨话题继续预训练 BERT 的表现（CDM）

话题\数据集	Laptops-14		Restaurants-14		Twitter	
	准确率(%)	F1 分数(%)	准确率(%)	F1 分数(%)	准确率(%)	F1 分数(%)
不继续预训练	**79.62**	76.01	84.11	76.44	73.70	72.37
Laptops	79.31	75.72	84.64	77.92	73.12	72.10
Restaurants	79.15	**76.42**	84.82	77.89	73.70	72.64
Twitter	78.84	74.79	**85.62**	78.22	**74.42**	**73.28**
ALL	79.31	75.70	85.09	**78.94**	73.55	72.34

表 3.8 话题内和跨话题继续预训练 BERT 的表现（CDW）

话题\数据集	Laptops-14		Restaurants-14		Twitter	
	准确率(%)	F1 分数(%)	准确率(%)	F1 分数(%)	准确率(%)	F1 分数(%)
不继续预训练	79.47	75.75	85.09	77.64	**74.42**	**73.72**
Laptops	78.68	74.37	84.46	76.28	72.11	70.89
Restaurants	78.84	75.76	85.98	79.34	73.27	72.26
Twitter	**80.09**	**76.13**	85.18	77.54	73.70	72.32
ALL	79.62	75.27	**86.25**	**79.66**	72.40	71.10

Sun 等人曾对任务内、领域内、跨领域语料继续预训练或微调 BERT 做了大量且详尽的实验，并在八个任务中都有了明显提升，取得了最先进的表现[25]。其中 Sun 等人总结了任务内语料继续预训练 BERT（BERT in-task further pre-training，BERT-ITPT）和领域内语料继续预训练 BERT（BERT in-domain further pre-training，BERT-IDPT）可以明显提高微调的表现[25]。其中，Sun 的实验过程表明，ITPT 在预训练约 100k steps 的时候效果达到了最优，之后会轻微变差[25]。在 IDPT 的实验中 IMDb 和 Yelp 数据集一个做继

续预训练数据，一个做微调数据，互相为彼此带来了比不继续预训练更差的效果，这是唯一发生变差的例子，Sun 等人表示原因可能是 IMDb 和 Yelp 数据集主题分别是电影和食物，二者的数据分布有着很大的差异[25]。因此本文在以通过继续预训练提高 LCF-BERT 性能为目标的同时，对 BERT-IDPT 是否能带来更好的表现是否与预训练数据集的话题有关进行了进一步的研究，即将 IDPT 划分为话题内继续语训练（in-topic further pre-training，IToPT）和跨话题继续预训练（cross-topic further pre-training，CToPT）。继续预训练分别使用的是表 3.4 中提及的 Laptops、Restaurants、Twitter、ALL 数据集，分别观察在 Laptops-14、Restaurants-14、Twitter 三个微调任务上 LCF-BRET$_{BASE}$ 的表现，结果如表 3.7、表 3.8 所示。

其中第一行是不继续预训练的表现，第二行到第四行的主对角线是 IToPT，第二行到第五行的其他部分是 CToPT。将在三个微调数据集上的准确率和 F1 分数共计视作 12 项评估。在这 12 列上，表 3.7、表 3.8 对最好的表现进行了加粗。其中 9 项通过继续预训练获得了更好的结果，准确率的最大提升约 1.5%，F1 分数的最大提升约 2.5%，这证明了 IDPT 确实有很大可能带来明显的提升。然而仍旧有三项不继续预训练反而更好，这证明了 IDPT 不一定能带来提升。其中，值得注意的是，Twitter 数据集我们并未扩充，而 CDW 结构下，Twitter 数据集不继续预训练反而准确率和 F1 分数最好，请注意 Twitter 数据集并未扩充，此时对 Twitter 的微调任务在 Twitter 数据集上继续预训练，不仅仅是 IDPT，还是 ITPT，而这个结果说明了即便是 ITPT 也未必带来更好的表现，这唯一的一个特例，与我们的常识和 Sun 等人的实验结果[25]是违背的。在通过 IDPT 取得最优的 9 项中，仅有 2 项是通过 IToPT 取得最优，其余 7 项均是通过 CToPT，在这 7 项中，仅有 1 项来自 Laptops 或 Restaurants 数据集的继续预训练，其余有 3 项来自 Twitter 数据集，有 3 项来自 ALL 数据集。另外可以观察到 Laptop 和 Restaurant 互为继续预训练数据集和微调数据集时，并没有带来多大提升，反而很大可能下降。根据以上实验结果，相比于 IToPT，CToPT 反而更容易带来提升。而 Laptop 和 Restaurant 之间的作用证明和话题或多或少还是有些关系。更进一步来说，实际上可以发现本实验中对微调表现带有决定性影响的实际上是继续预训练数据集分布的广度，因为带有提升的基本都是在 Twitter 和 ALL 数据集上的继续预训练。因此，可以推测，数据集的分布近似程度可能确实对继续预训练是否带来更好表现有着一定的影响，这个近似程度可能确实和话题有一定关系，而是否近似有多近似依然在理论上难以界定。同时在近似程度达到某一程度后，可能对表现的提升产生决定性作用的是用于继续预训练的数据集的分布广度，本实验中最后带来提升的基本都是分布较广的继续预训练数据集。另外，通过对领域语料继续预训练的研究与实验，也使得我在 Laptops-14 和 Restaurants-14 上取得了最先进的表现。

3.5 本章小结

本章详细介绍了了本文实验所用的数据集，并分析了其分布特点。又对超参数的设置进行了全面的阐述。对本文提出的方法以及复现的方法和过去研究者们方法的表现进行了对比，并分析了优缺点及产生的可能原因。之后对领域语料继续预训练 BERT 进一步划分为话题内和跨话题来研究话题是否是表现提升的决定性因素，在全面分析原因和总结结论的同时，也通过这一实验在其中的两个数据集上取得了最先进的表现。

4 案例分析

4.1 中性情感极性

Wang 等人曾在实验的案例分析中指出，在测试集上做错的很多都是情感极性为中性的样本，通过对案例语义的人工分析，指出预测错的样本往往预测的是整句的极性，对方面词的依赖不够，也因此指出情感极性为中性的样本可能更加依赖于方面词[7]。于是本文对本文实验中尝试的方法做错的案例做了进一步分析对比，如表 4.1、表 4.2 所示。

表 4.1 不同极性下错误数

方法	Laptops-14		Restaurants-14		Twitter	
	积极/消极	中性	积极/消极	中性	积极/消极	中性
TD-LSTM	97	99	106	153	161	46
ATAE-LSTM	89	95	108	145	137	84
TD-Transformer	138	78	155	135	135	67
TDLC-Transformer	102	96	143	121	132	63
TAGG-LSTM	104	101	152	109	113	84
LCF-BRET	54	73	71	83	93	84

表 4.2 不同极性下错误率（%）

方法	Laptops-14		Restaurants-14		Twitter	
	积极/消极	中性	积极/消极	中性	积极/消极	中性
TD-LSTM	20.68	58.58	11.47	78.06	46.53	13.29
ATAE-LSTM	18.98	56.21	11.69	73.98	39.60	24.28
TD-Transformer	29.42	46.15	16.77	68.88	39.02	19.36
TDLC-Transformer	21.75	56.80	15.48	61.73	38.15	18.21
TAGG-LSTM	22.17	59.76	16.45	55.61	32.66	24.28
LCF-BRET	11.51	43.20	7.68	42.35	26.88	24.28

其中使用的均为本实验尝试的方法，即 glove 词嵌入是 300d，其中 BERT 分别选取了三个数据集上结果表现最佳的方法，即对 laptops-14 微调，选取的是 twitter 数据集继续预训练得到的 LCF-BERT-CToPT-CDW，对 Restaurants-14 微调，选取的是 ALL 数据

集继续预训练得到的 LCF-BERT-CToPT-CDW，对 Twitter 微调选取的是不继续预训练的 LCF-BERT-CDW。

不同方法间一些较小的差距可以忽略不看，毕竟过程是有一定随机性的，结果会受到随机性的影响。本文是第一次全面拆分每个数据集逐一分析，从表 4.1、表 4.2 中，首先非常明显可以发现到的是，整体来看，中性的错误率是非常非常高的，总体来看，整个测试集上预测错误的几乎一大半都是在中性样本上。然而特别地，可以发现 twitter 数据集竟然中性样本预测错误率很低，这可能再一次印证了 3.3 节中分析的 twitter 数据集可能不是很依赖于方面词。整体来看，LCF-BERT 在不同极性下错误率均有明显下降，然而却在 twitter 数据集上中性样本的错误率达到了最高，同样在 twitter 中性样本上错误率最高的还有 ATAE-LSTM 和 TAGG-LSTM，ATAE-LSTM 是把方面词作为查询向量，TAGG-LSTM 和 LCF-BERT 都是突出局部语境，这三者都对方面词有着高度的依赖，这再一次印证了 twitter 数据集对方面词的依赖较弱，过度强调方面词的作用反而导致结果变差。而这样一个结果也再次说明了 TAGG-LSTM 的结构是有效的。

对于整体来看在中性样本上错误率非常高的问题，本文具体取出一些错误的例子进行分析，如表 4.3 所示。

表 4.3　预测错误的中性样本

例子	标签	预测值
okay soooo ... ummmmm .... what is going on with lindsay lohan’ s face? boring day at the office = perez and tomorrow overload. not good	0	-1
i heard ShannonBrown did his thing in the lakers game!! got ta love him	0	1

这两句中的方面词分别是 lindsay lohan 和 lakers。不难发现上述例子和 Tang 等人[6]、Wang 等人[7]的研究结论是一致的，错误原因是预测了整个句子的极性，对方面词的依赖不够。之所以中性样本错误多，正式因为中性样本对方面词依赖较高。然而过去的研究，以及上述例子，没有对例子来自哪个数据集进行分析。本文发现，这些例子基本都是来自于 Laptops-14 和 Restaurants-14。在 twitter 数据集上，恰恰相反，应当降低对方面词的依赖。基于方面的情感分析，提升对方面词的依赖显然应当是一个正确的思路。因此显然 twitter 数据集本身比较特殊，对方面词是一个弱依赖。这也解释了 3.3 节中，为何现在先进的方法在 Laptops-14 和 Restaurants-14 上的提升很大，然而却在 twitter 数据集上没有什么明显提升。我认为，在未来将 Laptops-14、Restaurants-14 数据集与 twitter

数据集分开讨论是一个合适的方向。另外，提出一个鲁棒性很强的、对方面词不同依赖程度都有很好表现的方法，将是未来的一个巨大挑战。

4.2 多方面词句子

在 ABSA 任务中，对同一句话中不同的方面词，可能得到的情感极性恰恰相反。考虑整句的情感不容易得到正确答案，答案来源往往聚焦在方面词附近。‘Price was higher when purchased on MAC when compared to price showing on PC when I bought this product .’这句话来自 Laptops-14 的测试集，句中的两个’price’均是方面词，对第一个’price’，情感极性是消极，对第二个’price’，情感极性是积极。从人类角度思考第一处的’higher’，价格高说明了情感是消极的，第二处的’price’，通过’when compared to’和前文’ higher’对比，可以发现情感极性是积极的。这是一个 LCF-BERT 全部做对，其他方法均未能全对的例子，图 4.1、4.2 展示了对第一个词的 CDM 和 CDW 处理，图 4.3、4.4 展示了对第二个词的 CDM、CDW 处理，不难发现，正是因为正确的聚焦，带来了正确的结果，毫无疑问 LCF 结构是有效的。

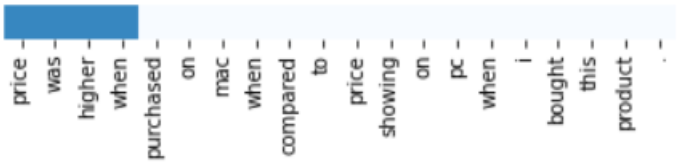

图 4.1 对第一个’price’的 CDM

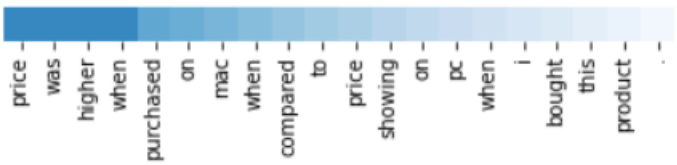

图 4.2 对第一个’price’的 CDW

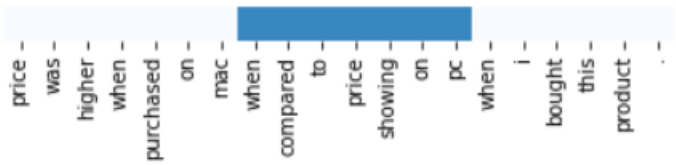

图 4.3 对第二个’price’的 CDM

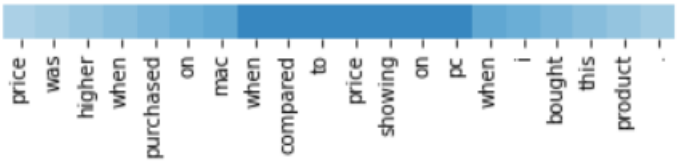

图 4.4　对第二个’price’的 CDW

4.3　本章小结

本章对不同情感极性样本的错误率进行了进一步分析。并指出主要错误来源是中性情感极性的样本，提高对方面词的依赖是解决措施。然而其中特例是 twitter 数据集对方面词依赖较弱。因此本文指出在未来将 Laptops-14、Restaurants-14 数据集与 twitter 数据集分开讨论是一个合适的手段。而提出一个适合全部数据集的鲁棒性强的方法也是未来的一个挑战。本章有对一句含有多方面词的例子进行了分析，通过对 CDM 和 CDW 的可视化指出了 LCF 结构的有效性。

5 系统落地

5.1 系统结构

考虑到投入使用的系统应当使用话题分布较为广泛的数据集训练出的模型，因此最后选择的是 twitter 数据集上表现最好的不继续预训练的 LCF-BERT-CDW。系统落地过程使用的是 Flask 框架构建来构建 web 应用。根目录下 app.py 文件起到了路由的作用连接前后端，实现了前后端的交互。templates 下是系统的页面 ABSA.html，在 static 下存放了对应的 css、png、js 等页面需要的静态资源文件。ABSA.html 通过 ajax 请求向后端发送与取回数据，展示情感极性与句子聚焦可视化热图。发送句子，以及方面词位置到后端。models 文件夹下存放了 LCF-BERT 的神经网络方法架构模型，state_dict 文件夹下存放了训练好的 LCF-BERT 的参数。app.py 同级下 data_utils.py 负责把输入序列转换为 token 并处理成 LCF-BERT 需要的数据格式，同级下 analysis.py 负责必要的设定输入 data_utils 处理好的数据预测情感极性以及准备可视化所需要的数据。ABSA.html 将数据发送到 app.py 后，由 app.py 调用 analysis.py 获得并返回预测值与可视化需要的数据。

5.2 系统界面

图 5.1 系统界面示例 1

系统由一个简易的界面构成，如图5.1、图5.2所示。其首部是大连理工大学的LOGO，然后是系统名称的标题。其中给定了一个本文框可以输入一个英文句子，又有两个下拉

选项框可以指定方面词的开始位置。在输入和选择前，框中均有对规格的提示。对句子长度或是方面词位置的输入不合理，也有正确的反馈。点击下方的情感分析 button 后，便会出现对情感极性的预测，以及通过 echarts 对整个句子聚焦的一个可视化。

图 5.2　系统界面示例 2

5.3　本章小结

本章主要介绍了系统落地使用的具体结构与运行流程。详细地介绍了项目文件的整体结构，每个包下所存放的文件，以及其所实现的具体功能，以及如何互相调用，按怎样的顺序调用的。并对系统界面进行了展示，解释了前端界面所含有的页面元素，及其功能与判断。

结　论

基于方面的情感分析是自然语言处理、情感分析中目前最热门的一项任务之一。本文通过对过去方法的全面剖析，对 LSTM、Tranformer 进行运用与结合，提出了了三个新的方法架构，TD-Transformer、TDLC-Transformer 和 TAGG-LSTM。实验证明 Transformer 可能并不是很适合处理序列性较强的任务，因为位置信息难以得到有效利用。而总体来看，TDLC-Transformer 的表现是还是不错的，LSTM 和 Transformer 等注意力模型相结合，或许是提高表现的一个不错的选择。

本文提出的 TAGG-LSTM 是一种有效的结构，然而网络规模较大，限于数据规模，在 Laptops-14、Restaurants-14 上表现并不好，而在 Twitter 数据集上有了更好的表现。未来，基于目标的神经元失活与聚合的方法有着进一步研究的价值。

在继续预训练 BERT 的实验中，本文发现，任务内语料继续预训练 BERT 也是存在不会对微调任务带来更好表现的可能的。领域语料继续预训练 BERT 确实有很大可能提高表现，话题可能是一种制约因素，但是本文发现，可能在预训练数据集与微调数据集分布近似程度达到一定程度时，对微调表现是否提高的主要因素可能是继续预训练数据集的分布广度。实验证明，话题内语料继续预训练带来提升的可能性并不高，合适的跨话题继续预训练反而可能带来很大的提升。

在案例分析中，本文再次证实了过去实验中提出的对中性情感极性样本的预测错误是主要错误来源，且相比于积极和消极，中性样本更加依赖于方面词，预测中出现的错误往往倾向于预测成整句的情感极性。然而本文的实验发现，Twitter 数据集是特殊的，所有的方法在 Twitter 数据集上，中性的预测表现很好，而积极和消极的预测表现反而差许多。本文指出，可能这是因为 Twitter 数据集对方面词有着弱依赖。本文认为，在未来将 Laptops-14、Restaurants-14 数据集与 twitter 数据集分开讨论是一个合适的方向。另外，提出一个适合每类数据集的、鲁棒性很强的方法，也是未来的一个挑战。

本系统最终实现了一个简易的系统落地，总体来说系统落地还有许多不足之处，功能上，指定方面词的位置对用户并不友好，未来可以进一步将其改造成使用端到端模型，自动完成从方面词抽取到基于方面的情感分类。另外，系统界面也有进一步美化的空间。同时，自己也要更多地了解各方面知识，并形成独到的理解。

参 考 文 献

[1] Pang B, Lee L. Opinion mining and sentiment analysis[J]. Foundations and Trends® in information retrieval, 2008, 2(1-2): 1-135.

[2] Jiang L, Yu M, Zhou M, et al. Target-dependent twitter sentiment classification[C]//Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies. 2011: 151-160.

[3] Dong L, Wei F, Tan C, et al. Adaptive recursive neural network for target-dependent twitter sentiment classification[C]//Proceedings of the 52nd annual meeting of the association for computational linguistics (volume 2: Short papers). 2014: 49-54.

[4] Zaremba W, Sutskever I, Vinyals O. Recurrent neural network regularization[J]. arXiv preprint arXiv:1409.2329, 2014.

[5] Hochreiter S, Schmidhuber J. Long short-term memory[J]. Neural computation, 1997, 9(8): 1735-1780.

[6] Tang D, Qin B, Feng X, et al. Effective LSTMs for target-dependent sentiment classification[J]. arXiv preprint arXiv:1512.01100, 2015.

[7] Wang Y, Huang M, Zhu X, et al. Attention-based LSTM for aspect-level sentiment classification[C]//Proceedings of the 2016 conference on empirical methods in natural language processing. 2016: 606-615.

[8] Ruder S, Ghaffari P, Breslin J G. A hierarchical model of reviews for aspect-based sentiment analysis[J]. arXiv preprint arXiv:1609.02745, 2016.

[9] Liu J, Zhang Y. Attention modeling for targeted sentiment[C]//Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers. 2017: 572-577.

[10] Chen P, Sun Z, Bing L, et al. Recurrent attention network on memory for aspect sentiment analysis[C]//Proceedings of the 2017 conference on empirical methods in natural language processing. 2017: 452-461.

[11] 孟霞. 基于层次化双向LSTM的评论方面级别情感分析研究[D]. 吉林大学, 2019.

[12] Xue W, Li T. Aspect based sentiment analysis with gated convolutional networks[J]. arXiv preprint arXiv:1805.07043, 2018.

[13] 程康鑫. 基于 LSTM 与 CNN 的中文餐饮评论情感特征提取算法[D].北京邮电大学,2020.DOI:10.26969/d.cnki.gbydu.2020.002649.

[14] Tang D, Qin B, Liu T. Aspect level sentiment classification with deep memory network[J]. arXiv preprint arXiv:1605.08900, 2016.

[15] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[J]. Advances in neural information processing systems, 2017, 30.

[16] Peters M , Neumann M , Iyyer M , et al. Deep Contextualized Word Representations[J]. 2018.

[17] Radford A, Narasimhan K, Salimans T, et al. Improving language understanding by generative pre-training[J]. 2018.

[18] 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.

[19] Song Y, Wang J, Jiang T, et al. Attentional encoder network for targeted sentiment classification[J]. arXiv preprint arXiv:1902.09314, 2019.

[20] Gao Z, Feng A, Song X, et al. Target-dependent sentiment classification with BERT[J]. Ieee Access, 2019, 7: 154290-154299.

[21] Zeng B, Yang H, Xu R, et al. Lcf: A local context focus mechanism for aspect-based sentiment classification[J]. Applied Sciences, 2019, 9(16): 3389.

[22] Rietzler A, Stabinger S, Opitz P, et al. Adapt or get left behind: Domain adaptation through bert language model finetuning for aspect-target sentiment classification[J]. arXiv preprint arXiv:1908.11860, 2019.

[23] Phan M H, Ogunbona P O. Modelling context and syntactical features for aspect-based sentiment analysis[C]//Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. 2020: 3211-3220.

[24] 王昆, 郑毅, 方书雅,等. 基于文本筛选和改进 BERT 的长文本方面级情感分析[J]. 计算机应用, 2020, 40(10):7.

[25] Sun C, Qiu X, Xu Y, et al. How to fine-tune bert for text classification?[C]//China national conference on Chinese computational linguistics. Springer, Cham, 2019: 194-206.

修改记录

（1） 毕业设计（论文）内容重要修改记录

第一次修改记录：

图片内容及位置修改，参考文献修改，**修改前：**图 1.1 中箭头太小看不清，图 1.1 和图 1.2 中字体太大，图 1.2 中背景色不合适导致部分字看不清。多处图片位置不合理导致大量空白。参考文献英文字体不正确。

修改后：重新绘制了图 1.1 和图 1.2，将其大小和颜色调整合适。所有图片位置进行了调整，减少了留白。参考文献英文字体修改为了宋体五号。

第二次修改记录：

图表位置修改，在各章之间插入了分页符，修改了英文关键字首字母的大小写，**修改前：**部分图表前后没有空行，各章间没分页，英文关键字首字母大小写不统一。

修改后：所有缺少空行的图表前后加了空行，各章之间加入了分页符，英文关键字首字母统一大写。

第三次修改记录：

标点符号修改，**修改前：**正文行文中同时存在英文和中文逗号、英文和中文括号。

修改后：正文行文中所有的逗号和括号都统一为了中文逗号和中文括号。

第四次修改记录：

图片中部分英文修改为中文，**修改前：**图 2.2、图 2.5、图 2.7 含有部分没有解释清楚且可翻译的英文术语，例如 mask、PE。

修改后：图 2.2、图 2.5、图 2.7 中部分可翻译的英文翻译为了中文。

（2） 毕业设计（论文）正式检测重复比：4.6%

记录人（签字）：宋安泽

指导教师（签字）：张晓彤

致　　谢

时光飞逝，岁月如梭，不知不觉中，大学四年的光阴早已飞逝而去。一路走来的点滴片段，此时此刻如同影像般在心底回放，无论快乐、忧伤、收获还是挫败，都是那样的珍贵与温暖。大学四年让我学习到了许多只有在大学才能学到的东西，拓宽了我的视野，充实了我的自己。这四年来结识了许多帮助了我很多的老师与朋友，大家一起交流，共同进步。大学，是一个终点，但大学，也是一个起点，以大学为始，我将用我所学，在日后的学习工作中砥砺前行。

首先，我要感谢我的母校大连理工大学，感谢学校的辛苦栽培，让我在四年内学到许多东西，逐渐成长。

其次，我要感谢我的毕业设计指导老师张晓彤老师，在我的毕业设计论文撰写过程中，耐心地指出格式等问题，并一一进行解答，再次表达对张晓彤老师的衷心感谢。

最后，我要感谢一直在背后默默支持我、关心我、爱护我的父母，感谢父母费心费力的抚养、无私奉献的帮助。在接下来的时间里，我将继续努力，来报答我的父母、我的家人。