西 南 交 通 大 学

本科毕业设计（论文）

基于序列标注的文本顺滑方法

年　　级：2019 级

学　　号：2019117649

姓　　名：王学彬

专　　业：自动化

指导教师：吕彪

二零二三年五月

西南交通大学

本科毕业设计（论文）学术诚信声明

本人郑重声明：所呈交的毕业设计（论文），是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

作者签名：

日期：　　年　　月　　日

西南交通大学

本科毕业设计（论文）版权使用授权书

本毕业设计（论文）作者同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权西南交通大学可以将本毕业设计（论文）的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本毕业设计（论文）。

保密□，在_____年解密后适用本授权书。

本论文属于

不保密☑。

（请在以上方框内打“✓”）

作者签名：　　指导教师签名：

日期：　　年　　月　　日　　日期：　　年　　月　　日

毕业设计（论文）任务书

班　　级 自动 2019-3 班　学生姓名 王学彬　学　　号 2019117649

发题日期：2022 年 12 月 19 日　　完成日期：2023 年 5 月 20 日

题　　目 基于序列标注的文本顺滑方法

1、本论文的目的、意义

自动语音识别（ASR）技术得到的文本中，总是含有大量不流畅现象。后续的自然语言理解系统（如机器翻译、句法分析等）因此会受到严重的干扰，因为这些系统往往建立在流畅的文本上。不流畅的现象主要是由两个方面导致的，一方面是语音识别技术的识别存在错误，另一方面是说话者的口语当中含有的语气词、停顿、纠正、重复等。文本顺滑任务的目的是解决后者带来的不流畅现象，经过处理得到顺滑的文本。本设计吸收、借鉴相关领域的研究成果，对于文本顺滑任务采用序列标注的方法。标签类型除了传统的标注词语是否为顺滑以外，尝试对于不顺滑词的标注采用更加多样的标签集合以充分利用语料中的标注数据，希望带来性能的提升。通过完成毕业设计，不仅有利于学生深化对专业知识的理解，而且有利于提高学生运用所学知识分析、解决学术问题的能力。

2、学生应完成的任务

(1) 明确研究目标和内容，学习文本顺滑、序列标注领域相关知识；

(2) 查阅文献并整理分析；

(3) 根据文本顺滑任务与语料中标签类型的特点，设计合理的序列标注 baseline 模型；

(4) 对序列标注 baseline 模型进行编程实现；

(5) 在序列标注 baseline 模型的基础上，结合实际语料标注情况尝试设计新的标签集合来获取更好的性能；

(6) 训练模型，保存模型；

(7) 完成论文撰写、外文资料翻译及相关文档的撰写。

3、本论文与本专业的培养目标达成度如何？（如在知识结构、能力结构、素质结构等方面有哪些有效的训练。）

本论文支撑本专业以下毕业要求的达成：（1）能够应用数学、自然科学和工程科学的基本原理，识别、表达、并通过文献研究分析复杂工程问题，以获得有效结论。（指标点 2.1-2.4）；（2）能够设计针对复杂工程问题的解决方案，设计铁道信号相关领域的系统、硬件部件和软件，并能够在设计环节中体现创新意识，考虑社会健康、安全、法律、文化以及环境等因素。（指标点 3.1-3.3）；（3）能够基于科学原理并采用科学方法对复杂工程问题进行研究，包括设计实验、分析与解释数据、并通过信息综合得到合理有序的结论（指标点 4.4）；（4）能够针对复杂问题，开发、选择与使用恰当的技术、资源、现代工程工具和信息技术工具，包括对复杂工程问题，尤其是铁道信号领域中复杂工程问题进行模拟仿真和数据分析（指标点 5.2）；（5）能够基于工程相关背景知识合理分析、评价专业工程实践和复杂工程问题解决方案对社会、健康、安全、法律以及文化的影响，并理解应承担的责任。（指标点 6.3）；（6）能够理解和评价针对复杂工程问题的工程实践对环境、社会可持续发展的影响（指标点 7.2-7.3）；（7）具备良好的口头表达能力和人际交往能力，能够就复杂工程问题与业界同行及社会公众进行有效沟通（指标点 10.1）；（8）具有书面表达与沟通能力、科技论文与技术报告写作能力，能够撰写报告和设计文稿、陈述发言、清晰表达或回应指令；（指标点 10.2）；（9）理解并掌握工程管理原理与经济决策方法，理解工程活动中涉及的重要经济与管理因素，并能在多学科环境中应用。（指标点 11.1-11.2）；（10）正确认识自我探索和学习的必要性和重要性，具有自主学习和终身学习的意识；（指标点 12.1）；（11）掌握正确的学习方法，具备自主学习能力，能够通过学习不断提高、适应信息技术的发展。（指标点 12.2）。

4、论文各部分内容及时间分配：（共 17 周）

第一部分 查阅并学习相关文献资料 (2 周)

第二部分 理清研究思路与基本框架，完成开题报告 (2 周)

第三部分 查阅文献资料，学习相关知识，根据文本顺滑任务与语料中标签特点设计合理多的序列标注 baseline 模型 (2 周)

第四部分 学习深度学习 PyTorch 框架，编程实现 baseline 模型 (2 周)

第五部分 在序列标注 baseline 模型的基础上，结合实际语料标注情况尝试设计新的标签集合来获取更好的性能，并且编程实现 (3 周)

第六部分 训练模型，进行实验，得出一定结论，检验是否符合预期 (3 周)

第六部分 论文撰写、外文资料翻译以及相关文档的打印与装订 (2 周)

评阅及答辩 (1 周)

备 注

指导教师： 2022 年 12 月 19 日

摘 要

人工智能的应用为我们带来了极大的便利，其中自动语音识别技术 ASR 尤其突出。然而，ASR 转录文本常常存在不流畅的现象，这会干扰下游自然语言处理系统的处理，如句法分析和机器翻译。为此，文本顺滑（亦称不流畅检测）任务应运而生，目的在于识别并去除转录文本中的不流畅部分，为下游提供更干净、顺畅的文本。在该任务的研究中，多数标签方案未能充分利用语料集的标注信息。因此，本毕业设计提出了一种细粒度标签方案，旨在通过充分利用标注结构信息来提升序列标注模型的性能。

本设计以过去研究采用的标签方案为基准，在语料集 English Switchboard 的标准注释结构基础上，设计了多种细粒度标签方案，比较不同粒度标签方案标注的语料数据对文本顺滑任务的影响。具体来说，本毕业设计首先阐述了在语料集 English Switchboard 的基准标签方案和嵌套细粒度标签方案的数据处理流程，而其余细粒度标签方案可以由嵌套细粒度标签方案转换而成。本毕业设计采用 BERT-CRF 序列标注模型，利用经过大量文本预训练的 BERT 获取良好的语义信息，CRF 层根据标签集对 BERT 层输出进行标注，通过组合两个性能卓越的模型，可获得比单一模型更佳的性能表现。最后，本设计在不同粒度标签的语料集上基于 BERT-CRF 模型进行了实验，对不同粒度标签集训练得到的模型进行了公平比较。实验结果显示，细粒度标签方案标注的语料数据训练的模型在测试集上表现出比基准方案更优越的性能，最佳细粒度标签模型相较于基线模型 F1 分数提升了 0.89。这表明细粒度标签方案能够为模型提供更多的标注结构信息，有助于模型识别语句中的不流畅成分。

本设计结合语料集标注结构，设计细粒度标签方案，并在 BERT-CRF 模型上与基准标签方案进行了对比实验。根据实验结果分析得出，细粒度标签方案相比于基准标签方案能充分利用标注结构信息从而提升模型的性能。

关键词：序列标注模型；文本顺滑；标签方案

Abstract

The application of artificial intelligence has brought us great convenience, particularly in the field of automatic speech recognition technology (ASR). However, ASR-transcribed texts often suffer from disfluency, which interferes with downstream natural language processing systems, such as syntactic analysis and machine translation. As a result, the text fluency (disfluency detection) task was created to identify and remove disfluency in transcribed text, providing cleaner and smoother text for downstream processing. In the study of this task, most labeling schemes fail to make full use of the annotation information of the corpus. Therefore, this thesis proposes a fine-grained labeling scheme aimed at boosting the performance of sequence labeling models by making full use of label structure information.

This thesis uses the labeling scheme adopted in previous studies as the baseline, and designs multiple fine-grained labeling schemes based on the annotation structure of the English Switchboard corpus to compare the impact of different granularity labeling schemes on text fluency task. Specifically, this thesis first expounds the data processing flow of the baseline labeling scheme and the nested fine-grained labeling scheme in the English Switchboard corpus, while the rest of the fine-grained labeling schemes can be converted from the nested fine-grained labeling scheme. This thesis adopts the BERT-CRF sequence labeling model, using BERT pre-trained with a large amount of text to obtain good semantic information. The CRF layer labels the output of the BERT layer according to the label set, and by combining two models with outstanding performance, the better performance can be obtained. Finally, this thesis conducts experiments based on the BERT-CRF model on corpora of different granularity labels and makes a fair comparison of the models trained with different granularity label sets. The experimental results show that the model trained on corpus data annotated by the fine-grained labeling scheme shows superior performance on the test set than the baseline scheme, and the best fine-grained labeling model improves the F1 score by 0.89 compared with the baseline model. This demonstrates that the fine-grained labeling scheme can provide the model with more annotation structure information, which helps the model to identify disfluent components in the sentence.

This thesis designs a fine-grained labeling scheme based on the corpus labeling structure and conducts a comparative experiment with the baseline labeling scheme on the BERT-CRF model. According to the analysis of experimental results, compared with the baseline labeling scheme, the fine-grained labeling scheme can make full use of the annotation structure

information to improve the performance of the model.

Keywords: Sequence Labeling model; Disfluency Detection; Label scheme

目 录

第 1 章 绪 论 1
1.1 背景与意义 1
1.2 国内外发展现状 2
1.3 论文所做工作及思路 4
1.4 论文章节安排 4
第 2 章 研究基础知识 6
2.1 自然语言处理概述 6
2.1.1 序列标注方法 7
2.2 统计机器学习模型 7
2.2.1 隐马尔科夫模型 7
2.2.2 条件随机场模型 8
2.3 深度学习研究基础 10
2.3.1 深度学习概念 10
2.4 本章小结 19
第 3 章 任务介绍和数据处理 20
3.1 文本顺滑任务 20
3.1.1 任务介绍 20
3.1.2 任务内涵 21
3.1.3 语料集介绍 22
3.2 数据处理 25
3.2.1 数据处理目标 25
3.2.2 思路介绍 26
3.3 本章小结 30
第 4 章 模型构建 31
4.1 序列标注模型 31
4.1.1 BERT 31
4.1.2 BERT-CRF 33
4.2 模型建立流程与数据预处理 35
4.2.1 模型建立流程 36

4.2.2 数据预处理 38
4.3 本章小结 42
第 5 章 实验与分析 43
5.1 模型训练与评估 43
5.1.1 模型训练细节 43
5.1.2 模型评估方式 45
5.2 实验及结果分析 46
5.2.1 训练结果 46
5.2.2 实验结果分析 48
5.3 本章小结 52
结 论 53
致 谢 54
参考文献 55

第 1 章 绪 论

1.1 背景与意义

移动互联网的高速发展有力地推动了人工智能领域的技术落地，便携式人工智能应用已经深入人们的生活，为我们带来了极大的便利。自动语音识别技术（Automatic Speech Recognition, ASR）是人工智能技术的重大应用，语音相比于文本而言，在日常生活中更加便捷。自动语音识别技术实现了语音到文本的转换，是连接人类语言与人工智能系统的桥梁，是实现人机自然语音交互的关键技术。而在自动语音识别技术得到的文本当中，总是存在着大量的不流畅现象，这种不流畅的现象导致语音识别技术转录得到的文本与书面文本存在较大的差异，对下游系统的处理产生不利的影响[1]。因为自然语言理解系统大多是基于流畅的、顺滑的文本进行构建与训练的，它们在含有不流畅的文本上的表现往往不尽人意，例如语音翻译、句法分析、文本摘要等。因此文本顺滑（也称作不流畅检测，Disfluency Detection）任务作为自动语音识别技术与下游的自然语言理解任务的关键链接，其重要性不言而喻。

不流畅的现象可以从两个因素加以分析，第一个因素是自动语音识别技术的不完全成熟导致在识别转录的文本中出现错误。但是随着技术的不断发展与进步，自动语音识别技术的错误识别率不断降低。另一个因素是说话者或讲述者的口语当中含有的语气词、停顿、纠正和重复冗余等不流畅语段，这一点是不顺滑现象的主要原因，同时也是文本顺滑任务聚焦的核心内容。文本顺滑的任务内涵是识别出文本中的不流畅现象，并将识别得到的不顺滑内容从原始文本中剔除，为后续的自然语言理解系统提供更加友好的文本。

本设计尝试利用自然语言处理领域较为成熟的序列标注方法处理文本顺滑任务。基于序列标注方法构建的模型在自然语言理解领域有着广泛的应用，许多基础的任务都是借此完成的，例如词性标注、命名实体识别、中文分词等。序列标注方法将输入文本视为一个序列，输出为根据输入文本序列与输出类别定义的等长的标签序列。在文本当中，上下文之间往往存在着相互依赖的关系，因此标签之间也存在依赖关系。序列标注任务并不是简单的分类任务，而是需要模型捕捉字词序列中的长距离依赖与层次化信息等，是一个结构化机器学习预测问题。序列标注模型方法的性能表现随着时代的发展获得了长足的提升，尤其是深度学习中的预训练模型极大地提高了序列标注模型的性能。

本设计具有一定的创新性与实用性，旨在探讨区别不同细粒度的标签集合对模型性能的影响，从标签方案集合的角度探讨文本顺滑任务模型的提升方法。在自动语音识别转录文本的基础上，期望通过设计特定的标签方案能够提升模型识别检测不流畅

语段的性能，为后续自然语言理解系统提供更加友好易用的转录文本。

1.2 国内外发展现状

序列标注是自然语言处理领域的一项基础任务[2]，其中涵盖词性标注（POS）、命名体识别（NER）和中文分词（CWS）等子任务，经过众多学者长期而充分的研究，在该任务领域涌现了大量优秀的方法与模型。

序列标注问题结合方法与特征，可以简要分为以下三类[3-5]：早期基于规则的方法、基于统计机器学习的方法和基于深度学习的方法。

早期研究中，基于规则的方法主要依赖于人类语言学家手工制定的复杂繁琐的规则[6]，在特定的领域具备着一定的准确率。但是这种方法的缺点也十分明显，往往模型的召回率不高，很难具备跨领域的迁移能力[3]。

基于统计机器学习的方法常常会设计特征模型得到文本序列的特征表示，因此也可称作基于特征模板的方法。基于特征模板的方法往往会根据领域特征或是语言特色精心设计特征模板[7, 8]，来得到文本序列的特征表示，然后利用机器学习算法，根据人工标注好的数据集进行训练。这种方法的模型通常具有较强的可移植性[3]，模型在不同的领域经过细微的修改或是设计新的特征模板再根据数据集重新训练即可得到较高的性能。常见的模型有：支持向量机（SVM[9]）、隐马尔科夫模型（HMM[10]）、最大熵模型（MaxEnt[11]）和条件随机场（CRF[12]）等。

基于深度学习的方法可以认为是机器学习的一个分支[13]，近年来，深度神经网络取得了空前的进步。反向传播算法和计算硬件性能的提升，使得训练更具规模和复杂度的深度学习模型变得可行。现今，针对序列标注任务，研究者们普遍使用的模型包括多种类型的循环神经网络（例如双向长短时记忆网络 Bi-LSTM[14]和门控循环单元网络 GRU[15]）、同时结合了卷积神经网络和条件随机场模型的模型（即 Bi-LSTM-CNNs-CRF[16]），以及预训练的语言模型（例如 BERT[17]）等。

在序列标注任务蓬勃发展的同时，文本顺滑任务也受到众多学者的广泛关注。在文本顺滑的研究领域，已经有过许多的学者尝试了非常多的方法与模型来提升文本顺滑任务的性能。

早在 1995 年，Shriberg 与 Metter 等提出了对文本中不顺滑的句子可行的标注规范方案[18, 19]，并且按照该规范标注的英文 English Switchboard（SWBD）语料集是文本顺滑任务一直使用的主要语料集，中文语言暂未公开大规模使用的语料集。Qian 和 Liu 在 2013 年提出一种不流畅检测的多步堆叠学习的模型[20]，从不相同的词序列中凝练提取出 n-gram 特征，再结合最大间隔马尔可夫网络（Max-Margin Markov Networks, M3Ns）达到了当时的最佳性能。除最大间隔马尔可夫模型以外，半马尔可夫模型（Semi-Markov CRF）以基于片段（span）的角度同样可以用于文本顺滑任务，如[21]

文献所述。2015 年，Wu 等人在传统的基于转移的依存句法分析模型中增加了 BCT 动作，以此融入句法信息，提升了文本顺滑模型的性能[22]。从技术上讲，将文本顺滑任务与句法分析结合起来是一个非常有效的解决方案。后来，神经网络的发展也为文本顺滑带来了新的高效模型。循环神经网络（Recurrent Neural Network, RNN）中的长短期记忆网络（Long Short-Term Memory network, LSTM[14]）有效解决梯度爆炸等问题，其网络特性在理论上是可以处理长距离依赖问题，因此也有研究人员将 LSTM 应用于顺滑任务，例如 Wang 等人于 2016 年提出使用人工特征的 LSTM-CRF 模型[23]来识别文本中的不流畅成分，并且取得了不错的结果；Bach 和 Huang 于 2019 年提出基于噪声 Bi-LSTM 的顺滑模型[24]，在结合 ELMo 与 GLoVe 嵌入词表示和自注意力机制等技术实现超越 BERT 的性能。

在文本顺滑领域，上述方法均受限于人工标注的数据集，后来研究人员提出半监督学习、自监督学习等方法减少对人工标注数据的依赖[25]。2020 年以来，Wang 等人提出了多任务自监督[26]、结合自训练和自监督[27]以及自适应无监督[28]的方法来减少对昂贵的人工标注数据的依赖，期望通过构造伪数据等数据增强手段来实现无监督模型，他们的模型也取得了非常不错的效果，在 SWBD 数据集上实现接近甚至超过现有监督模型的水平。其中，多任务自监督的工作，提出两个自监督任务：一是对输入文本序列的顺滑标注（Tagging）；二是句子级别的句子对分类任务（Sentence Classification）。前者训练目标为识别句子中的不顺滑词语，得到流畅的文本，通过构造大量的伪数据，来让模型学到尽可能多的信息；后者通过同时输入顺滑的句子与不顺滑的句子希望模型可以学到句子层次级别的句法信息。他们利用 Transformer 框架[29]进行预训练后再使用人工标注的数据微调，得到了非常不错的结果。

在可交互式的对话系统中，增量式（Incremental）文本顺滑任务更加关注于模型的实时性与低延迟，也是近年来一些学者的研究方向。2014 年，Honnibal 和 Johnson 提出了联合增量依存句法和文本顺滑任务的模型[30]，通过精心设计几类特征模板再基于一种新颖的增长式的、非单调的转移算法来处理顺滑标注中的最复杂的话语编辑（Edit）类型，并且经实验证明模型在两个任务上都取得了更好的效果。在追求更小更快的增量模型上，Chen 等人的工作[31]也值得探讨。这项工作提出一种流式的序列标注模型，通过训练模型学习动态调整前瞻窗口与等待时间来平衡准确性与延迟的方法实现实时的不顺滑词语的检测，且采用的 BERT 模型构架更小，也更适用于可移动式设备。

综上所述，文本顺滑这一任务从提出标注规范到如今，已经有了许多高效的模型与方法，无论是传统的机器学习模型，还是近几年出现的深度学习模型，在文本顺滑这一领域表现出的性能都十分出色，文本顺滑工作的准确率也在逐步提升。序列标注方法解决文本顺滑任务也是其中一个研究方向，有着众多学者在该方向进行探索，而

本设计旨在前人的基础上尝试对输入输出序列标签方案进行简单的改进，在现有高效的预训练模型基础上试验，期望模型通过细粒度标签方案的训练序列中能够学到更多的特征，从而提升模型的性能。

1.3 论文所做工作及思路

本论文的工作为利用序列标注方法尝试解决文本中不流畅现象识别检测的任务，并且根据不流畅检测任务的通用语料集的标准注释规范[18,19]，设计包含更多信息的细粒度标签方案，这是与目前主流的简单标签方案不同之处（作为本设计的 Baseline），也是本设计的创新之处。希望通过细粒度标签方案包含的丰富标注结构为序列标注模型提供更多不流畅现象的词汇、结构等信息，模型借此能够学习到更多的不流畅现象的特征。

这项工作主要按照如下四步进行：语料数据集的处理，构建模型，训练模型及实验，结果分析。

在语料数据集的处理中，会根据 English Switchboard 语料集的标注结构和标准注释规范对原始文本数据进行数据清洗工作，再依照不同的标签方案得到不同的文本词元（token）序列和标签序列，分别保存不同标签方案的数据集文件，防止后续混淆，并为模型的数据预处理做好准备。在这个过程中，数据清洗工作对于需要清洗的字符会存在难以统一编程实现、程序繁琐复杂的情况或运行效率低下的情况，可以采用正则表达式匹配目标字符，再进行后续操作。

构建模型涉及序列标注模型整体结构，模型的合理性探讨等。模型构建的过程中，需要掌握深度学习框架 PyTorch 的使用和相关的深度学习知识，否则在代码实现方面会存在困难。

训练模型与实验主要包括根据模型结构确定损失函数，选择合适的优化器加速训练模型并优化性能，设置超参数以及超参数调整等步骤。模型训练完成后还需要设计合理的评估方式或者评估指标来衡量模型在数据上的表现，确定不同模型间性能差异的判定方式。训练和实验之前还需要明确数据集中训练集、开发集和测试集的划分细节。

结果分析主要是对模型性能表现的总结和裁定，需要对实验结果进行详细分析和评估，并解释和讨论细粒度标签方案相比于 Baseline 对模型性能的影响和作用。

1.4 论文章节安排

论文共分五章。

第 1 章绪论。绪论包括背景与意义、国内外发展现状、论文所做工作及思路和论文章节安排。背景与意义部分介绍文本顺滑任务研究的来源和本毕业设计与现实相接

轨的实际意义。国内外发展现状小节简单介绍目前文本顺滑（不流畅检测）的研究情况。论文所做工作与思路小节讲述本论文的主要工作及实现思路。论文章节安排小节简介各章内容。

第 2 章研究基础知识。第 2 章从自然语言处理概述、统计机器学习模型和深度学习的基础知识等内容讲述本毕业设计需要的背景知识。统计机器学习模型简要以隐马尔可夫模型（HMM）和条件随机场模型（CRF）两个极具代表性的机器学习模型进行讲述。深度学习基础知识则围绕主要概念阐述，包括激活函数、损失函数、优化算法、评估指标等。

第 3 章任务介绍和数据处理。第 3 章第 1 节详细介绍文本顺滑任务及内涵，并且对使用的语料集和所用的标准注释结构进行了细致的阐述。第 2 节数据处理根据基准标签方案和细粒度嵌套标签方案介绍数据处理思路。

第 4 章模型构建。第 4 章核心目的是引入本毕业设计采用的序列标注模型 BERT-CRF，并介绍该模型建立流程和数据预处理等细节。

第 5 章模型训练、实验与分析。第 5 章主要关注不同标签方案的数据训练出的模型结果及模型评估方式，并对实验测试结果从模型性能、实际语料和存在的问题三个方面进行分析。

第 2 章 研究基础知识

2.1 自然语言处理概述

自然语言处理（Natural Language Processing, NLP）是计算机科学和人工智能领域中重要的分支之一，与计算机视觉（computer vision[33], CV）并驾齐驱。自然语言处理主要研究计算机如何处理人类语言，目标是使计算机能够理解、解读和处理人类语言，是沟通人类语言和机器语言之间的桥梁。

自然语言处理的两个核心任务是自然语言理解（Natural Language Understanding[32], NLU）和自然语言生成（Natural Language Generation[34], NLG）。具体来说，自然语言理解是指让计算机能够理解自然语言的含义，识别文本中的实体、情感、语义、逻辑等信息；自然语言生成则指的是让计算机自动根据给定的任务生成符合语言规范和语义逻辑的自然语言表达，以模拟人类的语言生成能力，如生成句子、段落、文章等。这两个任务的实现离不开数据驱动的技术，例如机器学习和深度学习，同时也要结合丰富的语言知识和规则，涉及到自然语言处理、计算语言学、语言学和心理学等多个学科的知识。这两个任务的研究和应用在很多领域都有着广泛的应用，例如智能搜索、情感分析、机器翻译、智能教育等。

自然语言处理包含多个子领域，各子领域之间存在着密切的关系和交叉点，相互之间的技术和方法也相互借鉴和发展。简要介绍一些 NLP 的子领域：

（1）语音识别[35]（Speech recognition）：将说话者的语音信号转换为文本表示。

（2）语音合成[36]（Speech synthesis）：将文本转化为合成音频，让计算机能够像人一样朗读出来。

（3）机器翻译[37]（Machine translation）：将一种语言的文本自动翻译成另一种语言。

（4）信息检索和文本分类（Information retrieval and text classification）：为用户提供文档、网页等信息资源的自动搜索和分类功能。

（5）序列标注[2]（Sequence Labeling）：确定文本中每个标记（如单词）的类别的任务，例如命名实体识别、词性标注等。

（6）情感分析[38]（Sentiment analysis）：分析文本中的情感极性，判断其中的情感倾向和态度，如正面、负面、中性等。

（7）问答系统（Question answering systems[39]）：可以回答自然语言提问的计算机程序。

（8）语言模型[40]（Language models）：用于把文本所包含的自然语言转换为计算机能够处理的格式。

2.1.1 序列标注方法

序列标注是自然语言处理领域中的一个重要子领域，具有十分广泛的应用，包括但不限于中文分词（Chinese word segmentation, CWS），词性标注（Part-of-speech tagging, POS），命名实体识别（Named Entity Recognition, NER），语义角色标注（Semantic Role Labeling, SRL）等 NLP 子问题都可以应用这种方法。

序列标注方法将输入文本视为一个序列，输出为根据输入文本序列与输出类别定义的等长的标签序列。每个标签都有特定的含义，是根据具体问题进行具体定义的，一般不能定义具有歧义的标签。在序列标注方法当中[41]，输入文本序列被称为“观测序列”，输出的标签序列称为“状态序列”。

在观测序列当中，标记（Token）之间往往存在着相互依赖的关系，因此标签之间也存在依赖关系。序列标注任务并不是简单的分类任务，而是一个结构化机器学习预测的复杂问题，需要模型捕捉标记序列中的长距离依赖与层次化结构信息等进行预测。序列标注方法的模型有基于传统的机器学习的统计模型，深度学习模型，迁移学习，半监督学习等。序列标注方法的准确率随着时代的发展获得了长足的提升，尤其是深度学习中的预训练模型极大地提升了序列标注模型的性能，这也是本设计所选择的模型方法。

在序列标注领域，统计机器学习模型和深度学习的模型都有广泛应用，接下来会对它们进行简要介绍。

2.2 统计机器学习模型

序列标注是一个历史悠久的问题，在统计机器学习领域中有经典的模型可供解决。特别是隐马尔可夫模型和条件随机场模型。机器学习是一种使用数据来学习预测函数的方法。机器学习算法主要是通过带有已知输出标签的训练数据来训练一个分类器或回归器，从而得到一个最优的函数。

2.2.1 隐马尔科夫模型

隐马尔可夫模型（Hidden Markov Model；缩写：HMM）亦可称作隐性马尔可夫模型，是一种统计模型。二元（一阶）隐马尔可夫模型是一种常用于序列标注任务的统计模型，其包括两个基本假设：第一个假设是，状态序列是一个马尔可夫过程，即当前状态只与前一个状态有关，与其它状态无关；第二个假设是，每个状态都有一个关联的观测值，但观测值并非直接观察得到，而是通过概率分布来描述，因此 HMM 也是一种概率图模型。基于这两个假设，HMM 模型将观测序列和状态序列联合建模，用来预测下一个状态。在序列标注任务中，HMM 模型可用于预测每个词性或命名实体类型的标注结果。

HMM 模型的参数包括初始状态分布、转移概率矩阵、发射概率矩阵。其中，初始状态分布指状态序列最初状态的概率分布；转移概率矩阵指从一个状态转移到另一个状态的概率；发射概率矩阵指从一个状态到达特定观测状态（即到达特定标记，如单词或字符）的概率。

HMM 模型有三个基本问题：评估问题，学习问题和解码问题。三个问题分别可以用三种算法进行求解：前向-后向算法，Baum-Welch 算法和维特比算法。

其中，评估问题是指在给定模型参数和观察概率的情况下，计算观测序列的概率值，前向-后向算法可用于这个问题的计算求解。

HMM 模型的学习问题是指给定一个包含许多观测序列的训练集，通过最大化观测序列在模型下的出现概率，来估计模型的参数。为了解决这个问题，通常使用基于 EM 算法的 Baum-Welch 算法来进行迭代求解。

维特比（Viterbi）算法可用于解码问题，是一种动态规划算法，用于计算观测序列对应的最可能的状态序列。

具体地，假设有一个 HMM 模型，参数为 $\lambda=(A,B,\pi)$ 和一个观测序列 $O=\{o_1,o_2,\ldots,o_T\}$，其中 A 是转移概率矩阵，B 是发射概率矩阵，π 为初始状态概率向量。

对于评估问题，计算观测序列 O 在给定模型 λ 条件下出现的概率。学习问题则利用 Baum-Welch 算法优化模型参数 A,B,π，使得该模型下观测序列的条件概率 $P(O|\lambda)$ 最大。解码问题也叫做预测问题，给定观测序列和模型，求解最可能的状态序列。

2.2.2 条件随机场模型

条件随机场（Conditional random field，CRF）是一种用于概率建模和分类的图模型，广泛应用于自然语言处理领域。CRF 是在 HMM 的基础上发展而来的一种无向图模型（undirected graphic model）。HMM 因为存在隐马尔可夫假设，t 时刻的状态仅依赖于前一时刻 $t-1$ 的状态以及各个时刻的观测之间条件独立，使模型在长距离依赖的建模上存在表现能力较弱的情况。而 CRF 没用这种限制，可以跨越不同位置之间的状态来建模状态之间的相关性或依赖性，使模型具有表达长距离依赖性和更灵活的特征和表达能力。实际情况中，大多数序列数据不具备观测序列的独立性假设，不能被简单表示为条件独立事件，因此 HMM 模型的性能会下降，而 CRF 则不会受到这样的限制，准确率明显更高。

2.2.2.1 CRF 模型表示

本章节以序列标注问题为例，介绍 CRF 模型及其应用的关键算法。

在 CRF 模型中，给定观测序列 x，求解最可能的状态序列 y^*，即 $y^*=\arg\max_{y\in Y}P(y|x)$。其中，y 表示一个标注序列，$Y$ 表示所有可能的标注序列的集

合。根据观测序列和标注序列，定义一个能够计算条件概率值的组合特征函数为：

$$f(x,y,i), y \in Y, 1 \le i \le n \tag{2-1}$$

其中，i 表示观测序列中的位置。特征函数 $f(x,y,i)$ 是定义在 x,y,i 的实值函数，函数取值表示在 x 序列 i 位置上呈现某种状态的概率大小。

CRF 模型定义联合概率如 2-2 式。

$$P(y|x) = \frac{1}{Z(x)} \exp(\sum_{i=1}^{n} w \cdot f(x, y_{i-1}, y_i, i)) \tag{2-2}$$

其中，w 是模型的参数；n 表示序列长度；$f(x, y_{i-1}, y_i, i)$ 是特征函数的一个例子；函数 Z 为配分函数；分母 $Z(x)$ 称为规范化因子，将所有的可能的状态序列概率求和为 1，可以通过一个动态规划算法（前向-后向算法）计算。

CRF 模型的训练过程可以使用 CRF-loss 函数对模型参数进行估计。简单的直观理解为，模型训练过程中通过优化参数 w 使得正确序列 y^* 在上述联合概率中趋近于 1，达到最大。模型的预测可以使用维特比算法求解，在所有可能的状态序列中找到最可能的一条状态序列路径。

2.2.2.2 维特比算法

维特比算法是常用的求解 CRF 模型与 HMM 模型最优解的方法，它的目标是在所有可能的状态序列中寻找可能性最大的状态序列。

维特比算法的核心思想是动态规划（Dynamic Programming），前文中提到的“前向-后向算法”也是一种动态规划算法。动态规划算法的核心思想是将一个复杂的问题分解为若干子问题，然后通过求解子问题得到原问题的解。维特比算法也是如此，将一个观测序列求解最优状态序列的问题分解为部分观测序列求解部分最优状态序列的子问题。动态规划算法在许多复杂的计算问题中都有广泛的应用，如最短路问题、最大流问题、DNA 序列比对、KMP 字符串匹配等。它的优点是可以解决许多非常复杂的问题，缺点是需要较多的空间来保存已经求解的子问题，同时也需要较长的计算时间。

给定观测序列，对于寻找最可能的状态序列问题，最容易想到的方法是通过暴力枚举得到所有可能的状态序列的概率，选择其中概率最大的作为最优状态序列。暴力枚举的思路是简洁的，但是其消耗的时间是难以接受的。枚举方法的时间复杂度为 $O(T^n)$，其中 T 表示状态的可能数量，在序列标注问题中为标签集大小，n 表示观测序列长度。该时间复杂度为指数级别，随着序列长度的增大消耗的时间增长非常迅速，这在实际应用过程中是无法接受的。

维特比算法的时间复杂度为 $O(nT^2)$。这个时间复杂度在多数情况下是较为合理的，因为标签集的大小并不会特别大。维特比算法流程如图 2-1。

Algorithm 1: Viterbi algorithm.

Input: Input sequence: $\boldsymbol{x} = \{x_1, x_2, ..., x_n\}$; the state space $\boldsymbol{S} = \{s_1, s_2, ..., s_T\}$ transition matrix $\boldsymbol{A}$ of size T, T, emission matrix $\boldsymbol{E}$ of size K, N, initial probabilities $\boldsymbol{\pi}$ of size K that π_i means the probability $x_1 == s_1$.

Output: The state sequence with maximum probability $\boldsymbol{y} = \{y_1, y_2..., y_n\}$.

Data structure: $dp[i, j]$ stores the probability of ; $traceback[i, j]$.

for $i = 1 \rightarrow K$ **do**
    $dp[i, 1] = \pi[i] * E[i, 1]$
    $traceback[i, 1] = 0$
for $i = 2 \rightarrow n$ **do**
    for $j = 0 \rightarrow K$ **do**
        $dp[j, i] = \max_t dp[t, i-1] * A[t, j] * E[j, i]$
        $traceback[j, i] = \arg\max_t dp[t, i-1] * A[t, j] * E[j, i]$
$z_n = \arg\max_t dp[t, n]$
$y_n = s_{z_n}$
for $i = n \rightarrow 2$ **do**
    $z_{i-1} = traceback[z_i, i]$
    $y_{i-1} = s_{z_{i-1}}$
Return $\boldsymbol{y}$.

图 2-1 维特比算法

相比于枚举解法，维特比算法是非常高效的，极大地减少了时间的消耗，并且使用动态规划的思想，考虑每个状态节点的局部最优解，通过局部最优解推导出全局最优解，具备良好的拓展性和可解释性。

2.3 深度学习研究基础

深度学习在自然语言处理中的应用也非常广泛。它能够帮助人们实现文本自动摘要、机器翻译、自然语言生成等任务。深度学习是一种建立在人工神经网络上的机器学习方法，可以构建多层次的网络来发现和利用数据中的模式和规律。深度学习方法通常需要大量的数据来训练模型，例如在自然语言处理领域中，训练深度学习模型需要大规模音频、文本和图像数据。这种方法的主要优势是它能够识别一些更为复杂、含有更高度层次或抽象特征的表达，比如语音识别和自然语言理解等任务。深度学习的网络有卷积神经网络、递归神经网络、生成对抗网络等。因此，深度学习模型需要大量的计算资源，对硬件要求颇高，这是在训练和部署方面都无可避免的问题。

2.3.1 深度学习概念

以下章节简要介绍深度学习中涉及到的关键技术和方法，如激活函数、损失函数、

优化算法、评估指标和深度学习框架等。

2.3.1.1 激活函数

激活函数（Activation Function）是神经网络中的重要组成部分，通常采用非线性函数帮助人工神经网络进行复杂的非线性化。在神经元中，通常神经元的输入经过一系列的加权求和操作后作用于某一函数，该函数即为激活函数。从字面意思理解，激活函数计算当前神经元是否被激活，决定是否将当前信息传递给下一层神经元。常用的激活函数有 Sigomid 函数、Tanh 函数、ReLU 函数和 LeakyReLU 函数。选择适当的激活函数有利于提高模型准确率。

（1）Sigomid 激活函数

Sigomid 函数也称为 Logistic 函数，输出范围在 0 到 1 之间，可以用于将输出解释为概率，主要用于二分类问题和输出层的概率预测，数学表示如 2-3 式。

$$f(x)=\frac{1}{1+e^{-x}} \tag{2-3}$$

其中，x为输入，$f(x)$为输出，e为自然常数。

函数图像如图 2-2 所示。

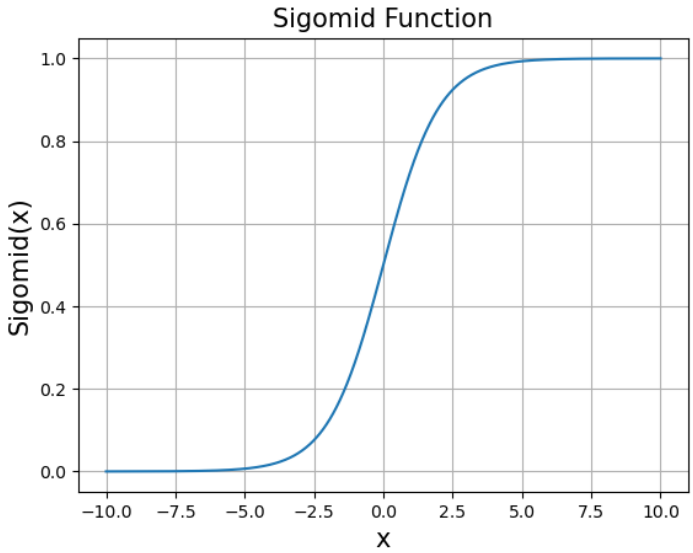


图 2-2 Sigomid 激活函数图像

Sigomid 函数的导数 $f'(x)$ 可以表示为：

$$f'(x)=f(x)(1-f(x)) \tag{2-4}$$

由 Sigomid 函数的导数可知，原函数处处可微，梯度平滑，导函数可以用于反向传播算法中计算梯度。Sigomid 函数的不足主要在于当输入为较大或较小值时，函数的导数趋近于 0，导致梯度消失，不利于神经网络的训练。在深度学习中，Sigomid 函数已经被更优秀的 ReLU 和 Tanh 激活函数取代。

（2）Tanh 激活函数

Tanh 函数全称为双曲正切激活函数（Hyperbolic Tangent Activation Function）也是一种常用的激活函数。定义为：

$$f(x)=\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{2-5}$$

Tanh 函数图像如图 2-3 所示。

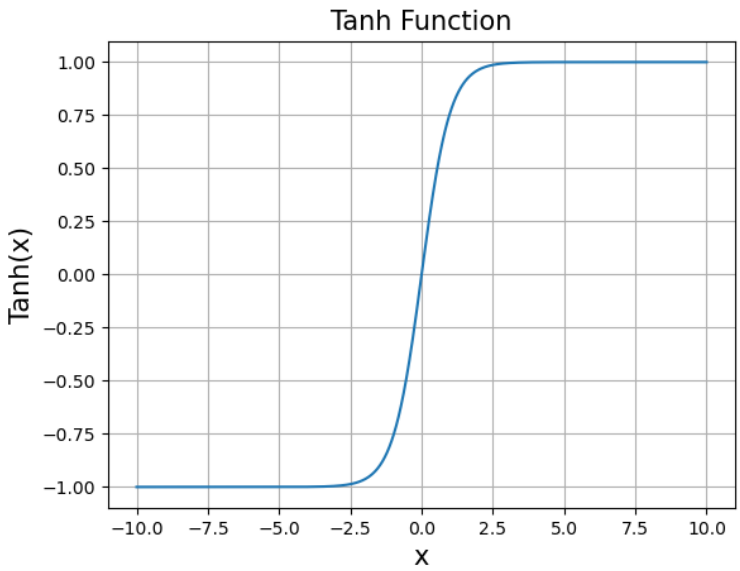


图 2-3 Tanh 激活函数图像

从图 2-3 可以看出，Tanh 函数与 Sigomid 函数较为类似，但是其输出范围为-1 到 1，且以 0 为中心。因此 Tanh 函数的输出范围更广，在一些情况下更适合作为输出函数。类似地，Tanh 函数同样存在 Sigomid 函数存在的梯度消失的问题，在使用 Tanh 函数时应该尽量注意输入值范围，避免这种问题。

（3）ReLU 激活函数

ReLU 激活函数全称为修正线性单元（Rectified Linear Unit），是分段线性函数，在目前的深度神经网络中被广泛使用。定义为：

$$f(x)=\begin{cases}x, & x>0\\ 0, & x\le 0\end{cases} \tag{2-6}$$

ReLU 函数图像如图 2-4 所示。从 ReLU 函数图像中可以看出：当输入大于等于 0 时，输出为线性关系；当输入小于 0 时，输出为 0，是非线性关系。因此大于等于 0 部分可以看作是一种线性激活函数，小于 0 部分可以看作是一种非线性激活函数。此外，当输入为 0 时，ReLU 激活函数的导数不存在，因此在此处不可微，在实际应用中，可以将其导数定义为 0 或 1。

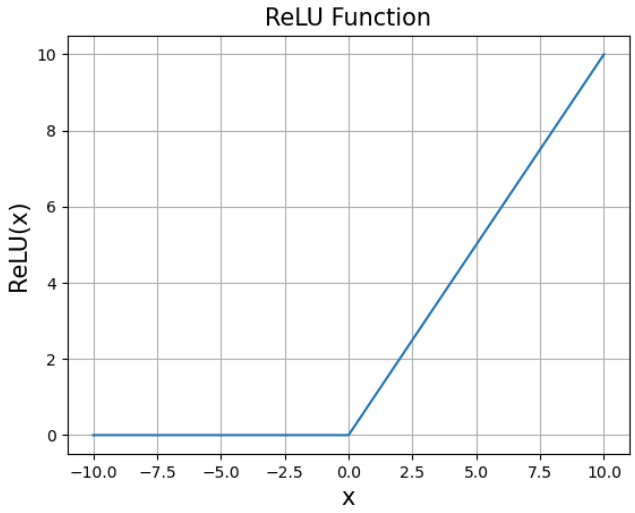


图 2-4 ReLU 激活函数图像

ReLU 函数简单有效，仅判断输入是否大于 0 即可。同时，有效缓解梯度消失问题，加速收敛。此外，ReLU 函数具备稀疏性的特点，可减少计算量和存储空间，因其在负半轴上输出为 0。

但是，ReLU 存在“神经元死亡”的问题。当输入小于等于 0 时，输出为 0，梯度也为 0，则无法利用反向传播更新权重。

2.3.1.2 损失函数

当深度学习模型训练时，使用损失函数衡量模型的发散程度。损失函数主要是用于模型训练阶段，模型根据批次输入进行前向传播计算预测值，损失函数计算预测值与真实值之间的差异，也就是损失值。损失函数的目标是尽量降低模型的损失，使模型学习到最优解。损失函数有多种类型，常用的有均方误差（Mean Squared Error，MSE），交叉熵损失（Cross-entropy loss）等。

（1）均方误差损失（Mean Squared Error，MSE）

均方误差损失又称为二次损失、L2 损失，常用于回归预测任务中。模型性能优劣可以通过 MSE 函数计算方差来衡量，预测值与真实值之间差距越小，则方差损失越小，说明模型预测能力更强。数学定义如 2-7 式。

$$loss_{MSE}=\frac{1}{n}\sum_{i=1}^{n}(y_i-\hat{y}_i)^2 \tag{2-7}$$

其中，n表示训练数据规模，y_i表示每个训练数据的输出真实值，$\hat{y}_i$表示模型对每个训练数据的预测值。

假设，真实值$y_i=100$，预测值$\hat{y}_i\in[-1000,1000]$。由此绘制的 MSE 函数图像如图 2-5 所示。

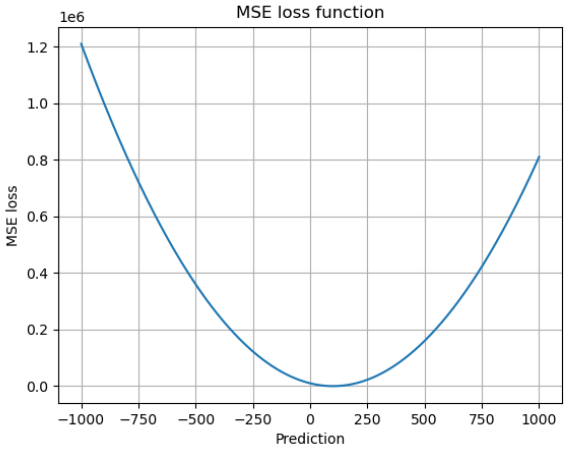


图 2-5 MSE 损失函数图像

从图 2-5 可以看出，当预测值越接近 100，MSE 损失就越接近 0，MSE 损失的范围为$[0,+\infty]$。

（2）交叉熵损失（Cross-entropy loss）

交叉熵损失函数，也称为对数损失或者 logistic 损失，常用于分类问题。函数定义为公式 2-8。

$$Loss_{CE}=-\frac{1}{n}\sum_{i}\sum_{c=1}^{M}y_{ic}\log(p_{ic}) \tag{2-8}$$

其中，M 表示类别数量；y_{ic} 取值 0 或 1，表示样本 i 类别为 c 取 1，否则取 0；p_{ic} 表示预测样本为 c 的概率，log 底数为 e。

在二分类问题中，上式可简化为公式 2-9。

$$Loss_{CE}=-\frac{1}{n}\sum_{i=1}^{n}y_i\log(p_i)+(1-y_i)\log(1-p_i) \tag{2-9}$$

以二分类问题绘制交叉熵损失函数图像如图 2-6（真实值为 1）。

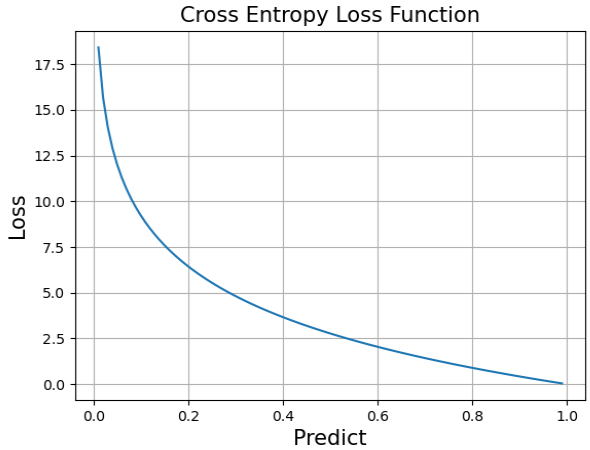


图 2-6 交叉熵损失函数图像

由图 2-6 可知，交叉熵损失函数是凸函数，具备良好的数学性质。交叉熵损失函数的主要思想是：模型预测越接近真实情况，则损失值越小；反之，模型的预测结果与实际结果不一致时，交叉熵损失函数值增加，以此调整模型参数，使模型预测更接近真实值。交叉熵损失函数能有效地处理多分类问题和数据不平衡问题，并且有效地避免梯度消失。因此交叉熵损失函数在神经网络中应用广泛。

2.3.1.3 优化算法

在深度学习模型中，优化算法（也可称为优化器）被用来迭代训练模型，更新调整模型的参数，使损失函数的值最小化，从而提升模型的预测能力。

在优化中，损失函数被称为优化问题的目标函数。此外，优化算法还需要包含另外几个必需的要素：梯度计算，学习率和参数更新。梯度是指损失函数对模型参数的偏导数，反映了损失函数在某个点上的变化率；学习率是指每次参数更新的步长，决定模型参数更新的速度和稳定性，学习率通常需要根据实际情况调整，以保证模型的训练效果；参数更新是指根据梯度和学习率来更新模型参数的过程，决定模型的训练效果和收敛速度。常见的优化算法有：随机梯度下降（stochastic gradient descent， SGD），自适应学习率优化算法（Adagrad，RMSprop，Adam，AdamW）等。

深度学习中，常用的优化方法是梯度下降。当目标函数为凸函数时，梯度下降法能得到全局解，但不能保证是最优解。假设在一个线性回归问题中，需要拟合的函数设为$h(x)$，采用均方误差损失函数设为$J(\theta)$，需要迭代求解模型参数θ得到拟合函数$h(x)$。给定m个样本的训练数据集，特征个数为n，。拟合函数和总的目标函数为公式 2-10 和 2-11。

$$h(\theta)=\sum_{j=1}^{n}\theta_j x_j \tag{2-10}$$

$$J(\theta)=\frac{1}{2m}\sum_{i=1}^{m}(y_i-h_\theta(x_i))^2 \tag{2-11}$$

给定学习率或步长α，模型参数更新如 2-12 公式所示。

$$\theta_j=\theta_j-\alpha\frac{\partial}{\partial\theta_j}J(\theta) \tag{2-12}$$

梯度下降算法有很多种变体，包括批量梯度下降法、随机梯度下降法和小批量梯度下降法。

除梯度下降算法外，介绍自适应学习率优化算法也是必要的。损失函数可能高度敏感于参数空间中的某些方向，则优化算法需要针对不同的参数设置不同的学习率。损失函数变化明显的参数方向（偏导数较大）：学习率较小，使得更新的步长较小。代价函数变化不明显的参数方向（偏导数较小）：学习率较大，使得更新的步长较大。

（1）随机梯度下降算法（SGD）

随机梯度下降算法与上述传统的梯度下降算法不同，每次只随机选择一个样本(x_i, y_i)进行梯度计算和参数更新。梯度下降算法时间复杂度为$O(n)$，随机梯度下降算法的的时间复杂度仅为$O(1)$，因此该算法降低了计算的时间复杂度，提高了模型的收敛速度。其参数更新公式如公式 2-13 所示。

$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta, x_i, y_i) \tag{2-13}$$

但是，SGD 算法也存在一些缺点。学习率α对算法性能影响较大，需要精心选择合适的学习率；对于非凸问题，由于仅选择部分样本的梯度进行计算，因此可能会陷入局部最优解。

（2）Adagrad 算法

Adagrad 算法是自适应学习率算法的一种。Adagrad 使用历史梯度平方和的平均值来调整学习率，使得梯度较大的参数更新较慢，梯度较小的参数更新较快。具体地，Adagrad 使用一个变量S来存储历史梯度平方和的平均值，每次更新时，学习率被除以S的平方根。这样可以使得学习率随着时间逐渐减小。数学表达式如式 2-14，2-15。

$$S = S + g \odot g \tag{2-14}$$

$$\theta_j = \theta_j - \frac{\delta}{\sqrt{S + \varepsilon}} \odot g \tag{2-15}$$

其中，δ为全局学习率，一个极小的常量ε保证分母不会为 0。一般情况下，全局学习率$\delta = 0.01$，极小常量$\varepsilon = 1e-8$。

（3）RMSprop 算法

RMSprop 算法是在 Adagrad 算法的基础上改进得到的，同时也是 Adam 算法的前身，关注于解决 Adagrad 算法中学习率逐渐衰减到 0 的问题。RMSprop 算法通过使用指数加权平均来计算历史梯度平方和的平均值，使学习率变化更加平稳。

具体地，RMSprop 使用一个变量S来存储历史梯度平方和的平均值，每次更新时，学习率被除以 S 的平方根。这样可使学习率在不同时间步长内自适应地调整，从而提高模型的性能。数学表达式如公式 2-16 和 2-17 所示。

$$S = \rho S + (1 - \rho) g \odot g \tag{2-16}$$

$$\theta_j = \theta_j - \frac{\delta}{\sqrt{S + \varepsilon}} \odot g \tag{2-17}$$

其中，ρ是衰减系数，通常设置为 0.9 或 0.99，用于平衡历史梯度平方和的平均值和当前梯度平方和的权重。其余参数与 Adagrad 算法一致。

（4）Adam 算法

Adam 算法是一个融合了动量梯度下降和自适应学习率的高效优化算法，通过计算梯度的一阶矩估计和二阶矩估计来自适应调整学习率。具体而言，Adam 算法针对

每个参数梯度的指数加权移动平均数（即一阶矩估计）和梯度的平方的指数加权移动平均数（即二阶矩估计）进行维护。

假设，t 为迭代次数，g_t 为当前迭代速度，θ_t 表示当前参数，δ 为学习率，m_t 和 v_t 分别表示当前梯度的一阶矩估计和二阶矩估计，β_1 和 β_2 分别为一阶矩和二阶矩的指数衰减率，经验值为 0.9 和 0.999。更新公式如 2-18 到 2-22 式所示。

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \tag{2-18}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \tag{2-19}$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \tag{2-20}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t} \tag{2-21}$$

$$\theta_{t+1} = \theta_t - \delta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon} \tag{2-22}$$

Adam 算法除了自适应调整学习率以外，还能有效处理稀疏梯度和非平稳目标函数，并且具备收敛速度快和性能良好的优点。

2.3.1.4 评估指标

在机器学习领域，模型的评估至关重要。评估策略有助于衡量模型的性能和可靠性，从而选择最优的模型或改进模型。

（1）混淆矩阵（Confusion matrix）

混淆矩阵是用来描述分类模型预测结果的矩阵，根据混淆矩阵可以得到 TP、FN、FP、TN 四个值，从而用来计算准确率、精确率、召回率和 F 值等指标。

混淆矩阵如图 2-7。

混淆矩阵		真实值	
		Positive	Negative
预测值	Positive	TP True Positive	FP False Positive
	Negative	FN False Negative	TN True Negative

图 2-7 混淆矩阵

其中，真正样本数 TP（True Positive）表示被模型预测为正的正样本；假真样本数 FP（False Positive）表示被模型预测为正的负样本；假负样本数 FN（False Negative）表示被模型预测为正的负样本；真负样本数 TN（True Negative）表示被模型预测为负的负样本。

（2）准确率（Accuracy）

分类问题中，准确率是最常用的评估指标。准确率表示分类正确的样本数占总样本数量的比例。计算公式如 2-23 式。

$$Accuracy = \frac{TP + TN}{FP + FN + FP + TN} \tag{2-23}$$

一般情况下，准确率越高，说明模型效果越好。但是，在样本数据严重不平衡的情况下，由于只考虑了分类正确的样本，准确率会失真。例如在文本顺滑任务中，不顺滑词只占全部词的小部分，若模型将全部词标注为顺滑，准确率依旧为一个较高的值，然而这样的模型并没有意义。

（2）精确率（Precision）

精确率也可称为查准率，是指模型预测为正样本的样本中，实际为正样本的比例。计算公式如 2-24 式。

$$P = \frac{TP}{TP + FP} \tag{2-24}$$

（3）召回率（Recall）

召回率又称查全率，表示模型正确识别出为正样本的数量占实际的正样本数量的比值。计算公式如 2-25 式。

$$R = \frac{TP}{TP + FN} \tag{2-25}$$

精确率和召回率都可以解决不同类别之间样本数量分布不平衡的问题。一般情况下，精确率与召回率越高，说明模型效果越好。但是，精确率和召回率是一对矛盾的度量值，总是存在精确率高而召回率低或召回率高而精确率低的情况，往往只在一些较为简单的任务中，精确率和召回率都很高。

（4）F 值（Fβ-Score 与 F1-Score）

F 值为精确率和召回率的调和平均数，可以综合评估模型的性能，F 值越高，说明模型的性能越好。计算公式如 2-26 式。

$$F\beta = (1 + \beta^2) \times \frac{P \times R}{\beta^2 \times P + R} \tag{2-26}$$

当 β 为 1 是说明对精确率和召回率的关注程度一致；当 β 小于 1 则更关注精确率；当 β 大于 1 则更关注召回率。在实际应用中，往往需要根据实际场景选择合适的 β 值，

但是常常取 $\beta=1$，即 F1 分数。

2.4 本章小结

本章主要就本毕业设计中涉及的理论研究知识或技术进行学习归纳。

简要介绍了计算机技术的一个重要研究方向——自然语言处理及其包含的子领域。重点介绍了在自然语言处理领域的子问题中的序列标注问题的经典统计机器学习模型：隐马尔科夫模型和条件随机场模型，并对 CRF 模型的解码算法——维特比算法进行了一定介绍。

此外，还简要介绍了深度学习和神经网络中一些常见概念：神经元的激活函数、损失函数、调整模型参数的优化算法和模型评估指标。

第 3 章 任务介绍和数据处理

3.1 文本顺滑任务

3.1.1 任务介绍

在移动互联网应用中，语音作为比文本更加便捷的信息源越发被大众所接受，自动语音识别技术随之被广泛利用。口语对话语言中往往存在不流畅现象，使语音转录得到的文本中同样包含不顺滑的表达，文本中的不流畅会导致后续任务的表现下降，比如机器翻译任务。因此文本顺滑任务（不流畅检测）是自动语音识别技术（ASR）连接下游自然语言处理任务系统不可或缺的一环，受到众多从事语音相关的研究人员重视[42]。

文本顺滑任务的目标是从输入的自然语言文本中检测出不流畅、不自然的部分，并对其进行标记。在后续处理中可以删除或修改润色被标记的文本，比如去除明显的错误、提供正确的词汇、语法建议等。

转录文本中不顺滑的部分包括以下几个方面：

（1）语义错误：文本中包含过度的重复、缺乏清晰的表达或表达不全面导致语义缺失等。

（2）拼写错误：这包括语音表达不清导致识别为错误的词语或是识别技术不完善导致识别错误的成分。比如，“womman”应该为“woman”。

（3）结构错误：包括语法错误、句子成分结构冗余等问题。该问题可以借助句法分析的知识处理。

（4）语气词、过渡词：在一些严格要求顺滑的场景中，语气词被认为是需要被识别的不流畅部分。但是在文本顺滑任务研究发展以来的主流方法并不把语气词或过渡词等视为不顺滑。

上述几个方面主要从语言学角度出发，可以看出文本顺滑任务的难点在于不流畅的表达方式可能因人而异即口语表述的差异化。因此，不流畅检测任务需要利用大量的语料库和自然语言处理技术，通过统计和分析不同情况下的文本表现，来自动判断哪些表达方式是不自然的。

文本顺滑技术在自然语言处理技术中应用广泛，从语音识别、机器翻译、自动问答系统到人机对话应用场景中，都需要解决这个问题。这个任务的成功可以极大地提高人工智能应用程序对自然语言的理解的准确性和可靠性，为人类提供更加便捷和智能化的交互体验。

以下章节将详细介绍文本顺滑任务的具体内涵。

3.1.2 任务内涵

口语表达与手写文本在许多方面都存在着较大的差异[43]。其中之一就是，说话者在对话的过程中会对部分之前说的话进行修正，抛弃或改换之前语段的意思，转而阐述新的语义。

从一个句子的结构和语义出发，口语对话转录文本的不流畅现象可以归纳为三类，分别是：填充停顿（filled pauses）、话语标记（discourse markers）和被编辑词（edited words）。

在对话文本中，填充停顿可以是无明显意义的语气助词或是无法有效转录的发声停顿，甚至可以不是一个完整的词。在中文中，这样的填充停顿常常指:“哎呦”、“嗯”、“啊”，“呃”等这类并没有实质语义的词语；在英文中可以是：“uh”、“hm”、“um”、“huh”等类似的语气填充词。

话语标记则不同于填充停顿，往往包含一定简单的语义并且表现出有意义的文段分明性，在话语层次中往往发挥着过渡、停顿或是转折等作用，因此在日常生活中几乎随处可见，并且各种语言中都广泛使用这样的词汇。在中文中，话语标记词包括：“然后”、“我的意思是”、“其实是”、“怎么说呢”等；在英文中，则包含：“I mean”、“because”、“you know”等。

被编辑词完全不同于填充停顿和话语标记，被编辑词一般含有明确的语义，或是不必要的重复。在语段中，被编辑词是说话者、讲述者（Speaker）表达错误的部分或是表达不明确的词，这一类词语使句意偏离了说话者想表达的含义或是在紧跟着的语段中说话者意识到之前的语句出现了事实上的偏差。因此，在完整语义的句子中，被编辑词后会紧跟着修复词（Repair）部分对语义进行纠正。如图 3-1 中文文本中的“去重庆”和英文文本中的“to Boston”。

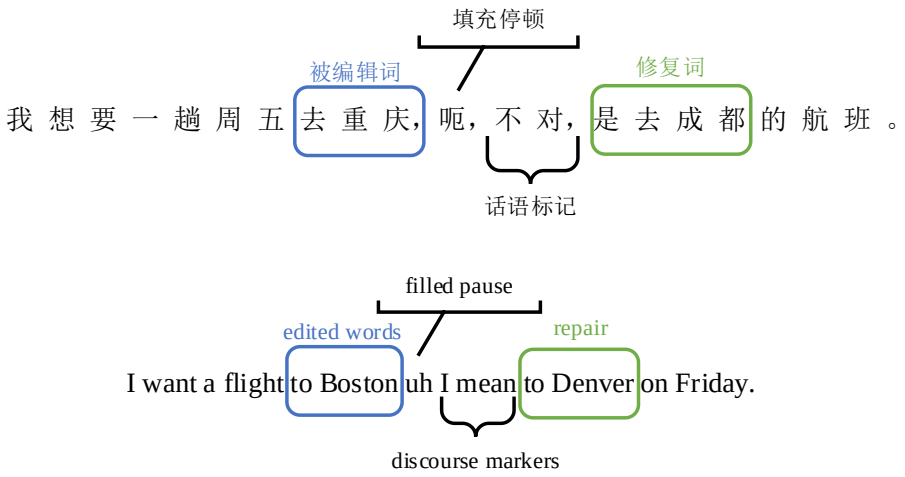


图 3-1 不顺滑现象中英举例

从图 3-1 可以更加直观地理解不流畅、不顺滑的句子所包含的成分。在这当中，

文本顺滑任务的内涵为识别并且标记其中的不顺滑成分——被编辑词。不流畅部分十分复杂，因为它可以是任意形式的短语。被编辑词长度是不固定的，出现的位置同样难以确定。而填充停顿和话语标记部分，相较于被编辑词部分来说较为简单，因为这两个结构往往是由封闭的短语集合组成的，通过设计特定的规则或是机器学习模型识别方式即能达到不错的效果。因此，在过去的大多数研究当中，研究者更加关注于被编辑词的检测，因此本设计也遵守主流研究方式的设定。

在文本顺滑任务当中，并没有主流的中文语料数据集，因此下面的篇幅仅介绍英文数据集。

3.1.3 语料集介绍

目前，在文本顺滑方法的研究当中，使用最为广泛的是 English Switchboard 语料集，以下简称 SWBD 语料集。SWBD 语料集由语言数据联盟（Linguistic Data Consortium，LDC）发布。

SWBD 数据集由两个人之间关于某个特定话题的电话录音转录得到。由于它是真实对话得到，因此其中包含的信息更加贴近现实情况，比如单词片段（不完整的单词）、中断停顿、不完整的句子和话语标记等内容。鉴于上述内容的繁多复杂，SWBD 语料集的注释由 Meteer 等人于 1995 年标注完成[19]，所采用的注释规则是 Shriberg 于 1994 年提出的不流畅注释标准结构[18]。正如前文所提及的，语料包含两个对话者之间的对话，以下分别指定为 speak A（A:）和 speaker B（B:），一个语料集中简短的对话如图 3-2 所示。

```
A.1:  All right, have you lived in this area long? /

B.2:  No, / I've only been here a couple of years. /

A.3:  Okay. /

B.4:  {F Uh, } {C but, } {F uh, } [ I've, + I've ] noticed that, {F uh, } this year
it seems to have been a lot warmer than it has in the past. /

A.5: Yeah, / [ [ [ it, + it, ] + it's, ] + it's ] a lot warmer, a lot drier too,
because usually about this time of year, [ this, + ] {D you know, } you see a little
bit more rain. / {E I mean } [ by, + by ] mid February, {E I mean,} we start getting
a lot of rain. /

B.6:  Yeah, / yeah, / {D well } I remember last year, {E or the year before, }

A.7:  {C But } it, -/

B.8:  {F uh, } we had ice and snow, {F uh, } {F uh, } / terrible ice storm, {F uh, }
around this time
of the year. /
```

图 3-2 SWBD 语料集部分对话

上图所示为目前 LDC 发布的语料集中的实际内容。其中的对话转录文档添加了

注释结构，因此并不能直接在模型中使用，需要对注释结构进行分离，得到“干净”的原始文本和标注结果。以此目的出发，详细了解数据集的不流畅语句的标准注释结构和符号含义是非常必要的，以下将详细展开介绍。

原始文本如图 3-3 所示，可以看出，B.6 语段与 B.8 语段进行了合并，因为 B.6 与 B.8 语段并非完整的句子，为保证模型输入的合法性与真实性，应尽量以完整的句子作为输入。显然，本来 A.7 语段也并不完整，但是结合 A.9 语段（仅为“Uh-huh.”）进行了补全。

```
A.1:  All right, have you lived in this area long?

B.2:  No, I've only been here a couple of years.

A.3:  Okay.

B.4:  Uh, but, uh, I've, I've noticed that, uh, this year it seems to have been a lot
warmer than it has in the past.

A.5: Yeah, it, it, it's, it's a lot warmer, a lot drier too, because usually about
this time of year, this, you know, you see a little bit more rain. I mean by, by mid
February, I mean, we start getting a lot of rain.

B.6:  Yeah, yeah, well I remember last year, or the year before, uh, we had ice and
snow, uh, uh, terrible ice storm, uh, around this time of the year.

A.7:  But it, Uh-huh.
```

图 3-3 语料集原始文本

3.1.3.1 标准注释结构

不顺滑语段的注释标准的组成与上述 3.1.2 中提到的不流畅分类存在着一定的相似性，但是并非完全相同。标准注释更加规范合理，具备系统性结构使得语料处理存在逻辑性，便于设计合理的算法，更易让人理解把握。

以下以图 3-1 中英文内容进行举例，标准不流畅注释结构如图 3-4 所示。

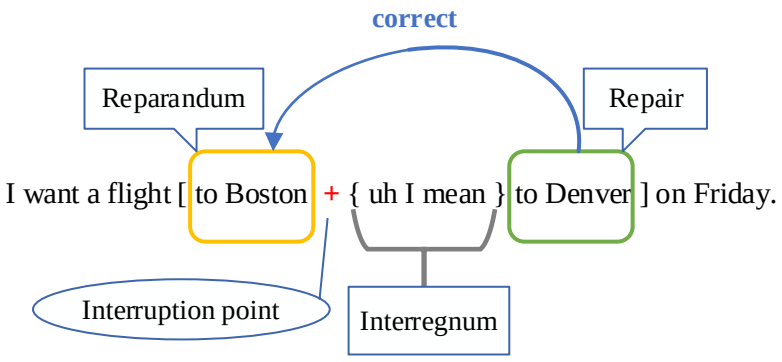


图 3-4 标准不流畅注释结构示例

一个标准的不流畅注释可包含四个部分：reparandum、interruption point、interregnum 和 repair。Reparandum 可翻译为撤销，Reparandum 部分是说话者想要忽略或者替代的词或词组。Interruption point 直译为中断点，是 reparandum 撤销词组的结尾标识符，表明 reparandum 的结束。Interregnum 译为中间词或者过渡词，这部分是灵活可选的，即并非一定会出现在一个不流畅语句的注释中，包括话语标记和填充停顿等。Repair 是与 reparandum 相关的修复词部分，提供代替与填充 reparandum 的新语义，修复表述者想要表达的语义。

[reparandum + { interregnum } repair]

图 3-5 标准不流畅注释结构

简要来说，不流畅注释结构如上图所示。这与前文提及的不流畅分类较为类似，但是通过人为添加中断点，使结构更具标注性，便于编程处理。Repair 修复词部分纳入注释结构考虑，使注释结构的逻辑性和系统性更强。此外，将填充停顿和话语标记等包含语义信息较少的词统一归为 interregnum，简化了注释内容但又不失结构的严谨性。

丰富的注释内容可为研究者提供许多信息，本毕业设计便是借助于注释结构对语料处理，设计不同的标签方案，建立序列标注模型，实验模型在不同标注方案下的性能表现，探讨标签方案对模型的影响。

3.1.3.2 不流畅类型

在标准不流畅注释结构中，interregnum 是不定可选的，并且存在于较为封闭的短语集合中，检测识别简单容易，且准确率非常高。Reparandum 与被编辑词等价，具备长度不定、出现不具备明显特征等特点，使模型较难学习到高效的特征，是文本顺滑任务的核心难点。

因此，在实际研究中，大多数研究者都会简化讨论范围，仅仅考虑模型对 reparandum 的识别情况。忽略 interregnum，不顺滑类型可以分为三种[45]：重复（repetition）、重启（restart）和修正（correction）。分类的标准是基于与 reparandum 关联的 repair 情况进行分类。若 repair 为空，分类为重启；若 repair 与 reparandum 内容一致，则分类为重复；若 repair 与 reparandum 内容不一致，则分类为修正。

表 3-1 不顺滑类型

repair 情况	**类型**
repair 为空	重启（restart）
repair 与 reparandum 一致	重复（repetition）
repair 与 reparandum 不一致	修正（correction）

举例如表 3-2 所示，其中例子均删掉 interregnum 以便阅读。

表 3-2 不顺滑类型举例

类型	注释举例
restart	[we would like +] let's go to the…
restart	[this, +] you see a little bit more rain.
repetitions	In summer [I, + I] like one piece dresses.
repetitions	[I've, + I've] noticed that…
corrections	[I used to, + my wife and I we used to] live in San Antonio.
corrections	[it was just, + it was] horrible.

3.2 数据处理

在 3.1 节中详细讲解了 SWBD 语料集的标准注释结构，为处理语料奠定了基础。SWBD 中的语段并不能直接输入到模型中使用，因此需要将语料处理为模型可用的输入输出形式，这个过程称为“数据处理”。

3.2.1 数据处理目标

在对数据进行处理之前，首先需要明白数据处理的目标，即明确“将语料处理成什么样的数据”。在序列标注模型当中，模型的输入为文本序列，模型的输出为等长的标签序列。例如，在中文词性标注任务当中，给定分好词的观测序列，模型的输出为等长的状态序列，如表 3-3 所示。

表 3-3 观测序列和状态序列（词性标签）举例

索引	观测序列	状态序列（词性标签）
0	戴相龙	NR
1	说	VV
2	中国	NR
3	经济	NN
4	发展	NN
5	为	P
6	亚洲	NR
7	做出	VV
8	积极	JJ
9	贡献	NN

在本毕业设计任务当中，因为采用的是序列标注模型，所以数据处理的目标类似于上表所示，但是因标签方案的不同会存在差异。标签方案可以简要分为基准标签方

案（baseline）和细粒度标签方案（fine-grained），在后续实验的过程中，细粒度标签方案包含多种不同的细粒度标签方案，与 baseline 进行比较。

3.2.2 思路介绍

在正式介绍如何处理数据之前，需要引入一个简单的概念——嵌套注释结构。如 3.1.3.1 节中所述的注释结构仅为非嵌套标准注释结构。从结构上理解，嵌套注释结构是在非嵌套注释结构中任意位置插入新的非嵌套标准注释结构；从语义上讲，嵌套注释结构的存在是由于在长难句中存在多次复杂不流畅、不顺滑的情况，或是不顺滑的词又夹杂顺滑的信息，就会使得该句的不流畅结构变得复杂。

在 SWBD 数据集中，大多非顺滑句子为非嵌套结构的不流畅注释，仅有少部分非顺滑句子为嵌套结构。嵌套标签的语料句子数量仅占整个数据集语料数量的 3.71%。

在本节的介绍中，给出两种标签方案作为代表：baseline 标签方案和嵌套结构细粒度标签方案。Baseline 标签方案是最为简单的方式，而考虑嵌套结构的细粒度标签方案则为在本毕业设计中最为复杂的处理。给定一个标注语料，如表 3-4 所示。

表 3-4 嵌套注释语料

[[I've, + I've heard some,] + {E I mean } I've heard [sta-, + statistics]] and things.

经过处理，得到的两种标签方案结果如表 3-5 所示。

表 3-5 不同标签方案标注结果

句子	baseline	嵌套方案
I've,	D	B-RM | S-RM
I've	D	M-RM | B-RP
heard	D	M-RM | M-RP
some,	D	E-RM | E-RP
i	O	B-IM
mean	O	E-IM
I've	O	B-RP
heard	O	M-RP
sta-,	D	M-RP | S-RM
statistics	O	E-RP | S-RP
and	O	O
things	O	O

在上表的结果中，可以看出 baseline 的标签集为{O, D}，其中 O 表示顺滑，D 表示不顺滑。在嵌套标签方案中稍显复杂，RM 为 reparandum 缩写，IM 为 interregnum 缩写，RP 为 repair 缩写。前缀标签集合{B, M, E, S}是对后缀标签的修饰，分别表示

的含义如表 3-6 所示。

表 3-6 前缀标签含义

前缀标签	含义
B	首位
M	中间
E	末尾
S	单独一个词

例如，“B-RM”表示 reparandum 的第一个词元；“S-RP”表示 repair 有且仅有一个词元；“M-IM”表示为既不在 IM 起始也不在末尾的中间部分词元。此外，表 3-5 中“|”表示不同层级标签的连接符。

3.2.2.1 Baseline 标签方案

假设已得到一句不流畅的标注语料，且注释结构完整，记为S，长度为n。其中S包括注释字符，因此S并非真实语句，n并非句子的实际长度。

根据以往学者的研究结果，本设计采用的 baseline 标签方案将只将 RM 部分作为非顺滑部分，标记为 D，而其余部分均可被视为顺滑部分，标记为 O。

在标准注释结构中，RM 部分的标记通常被放置在注释符号“[”与“+”之间。此外，另一个重要的细节在 baseline 标签方案的处理中发挥作用——即注释符号“[”与“+”之间的内容，无论其注释结构如何嵌套，都应视为非顺滑成分。因此，baseline 标签方案不考虑嵌套结构的细节处理。这种标签方案的使用简单，有助于更好地理解文本中的顺滑和非顺滑成分，并为后续的分析和应用提供基础。

综上所示，对于 baseline 标签方案而言，在处理非顺滑的语句时，仅需在遍历包含注释符号的标注语句时借助一个栈记录注释标志字符，通过栈的状态直接判断后续读取的 token 是否顺滑即能完成处理。

各注释符号含义如表 3-7 所示。

表 3-7 各注释符号含义

标志字符	含义
“[”、“]”	RM
“{”、“}”	IM
“+”	RP

处理流程图如图 3-6 所示。处理流程简要概括如下：在遍历标注语句时，当读入“[”字符时，将这个字符压入栈，在后续读取到 token 时，因栈内含有“[”字符则栈不为空栈，所以 token 都被标记为 D。当遇到“+”字符时，执行出栈操作（即将栈内的“[”弹出），此时栈为空栈，空栈时读取到的 token 都应标记为 O。

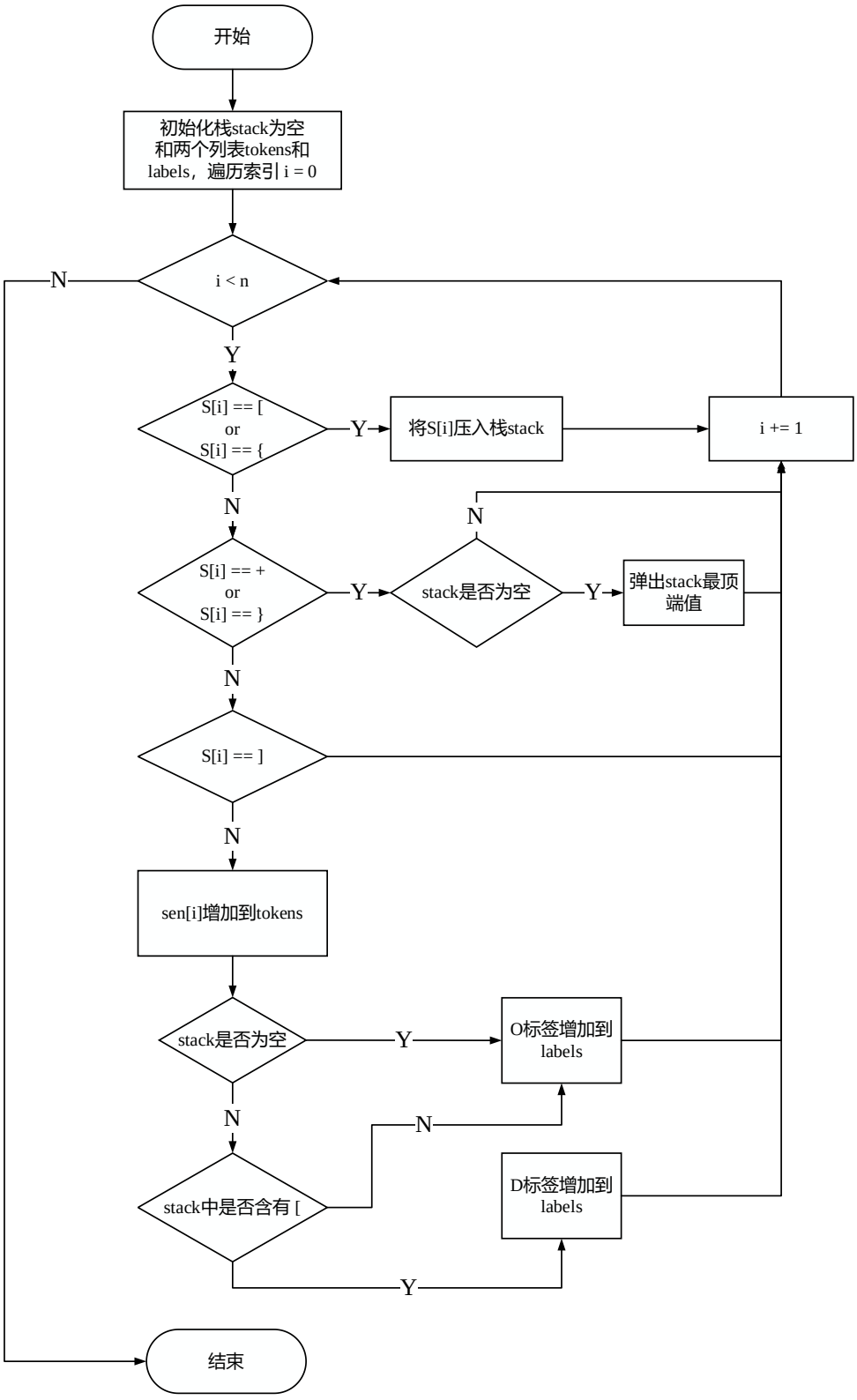


图 3-6 baseline 标签方案处理流程图

经上述流程后，最终得到一句不流畅的标注语料分离的词元序列 tokens 和标签序列 labels。

3.2.2.2 细粒度嵌套标签方案

在细粒度非嵌套标签方案中，需要同时考虑前缀标签{B, M, E, S}和后缀标签{RM, IM, RP}，前缀标签指示 token 位置信息，后缀标签明确 token 所属类别。而在嵌套标签方案中还需要使用连接符“|”得到每一个 token 详细标签结果。

嵌套标签方案的处理与 baseline 标签方案的处理思路是不相同的，但是非嵌套标签方案的处理与 baseline 标签方案是类似的，因此在 baseline 的基础上稍微改进可以得到非嵌套标签处理函数。

嵌套标签方案处理思路简要为：顺序遍历一句注释语料，在读取到一个完整的标准注释结构，视为一个片段。将该片段以非嵌套标签函数进行处理，完成处理后返回非嵌套标签并按位保存，删除该片段所使用的注释标志字符。循环上述方式直到整句注释语料不含注释标志字符，说明处理完成。

上述处理方式必须要求注释语料结构完整，不存在缺陷结构。流程图如图 3-7。

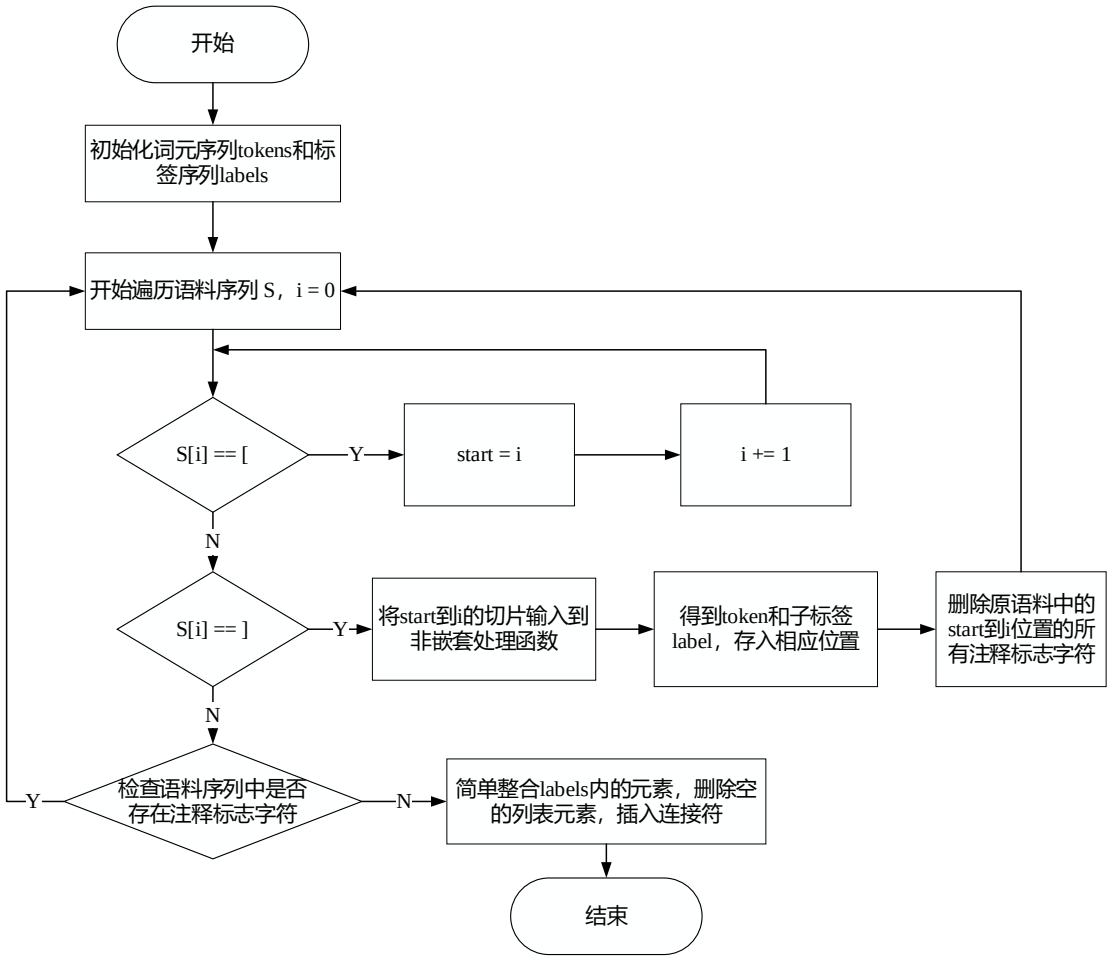


图 3-7 嵌套标签方案流程图

初始化时，labels 其内的元素为空列表。得到最终结果之前需要简单处理：labels 序列将空列表删除，将非空列表插入连接符变为字符串元素；tokens 序列也需删除对

应位置的元素。最终，得到等长的词元序列 tokens 和嵌套标签序列 labels。

3.3 本章小结

本章 3.1 节详细介绍了文本顺滑任务——检测与标记给定语句的非顺滑部分，并且详细介绍了 SWBD 语料集及使用的标准注释结构。

3.2 节详细讲述了按照两种标签方案处理语料集的思路与流程。Baseline 标签方案简单有效，将文本顺滑任务视为二分类问题，并不考虑多余的结构信息；细粒度标签方案充分利用标注结构的信息，期望通过更细粒度的标签方案相较于 baseline 标签方案让模型在训练过程中学到更多的特征，以此提升模型的性能。这个目标是本毕业设计的出发点。

经本章任务介绍与数据处理步骤后，明确了任务需求和目标，得到了序列标注模型的标准输入输出，为后续模型的训练和评估等工作做铺垫。

第 4 章 模型构建

4.1 序列标注模型

本毕业设计所采用的序列标注模型为 BERT-CRF。BERT 是 Google 发布的预训练模型，可用于序列标注任务，且表现优异。CRF 是传统机器学习中的经典代表，在序列标注任务中一直扮演着举足轻重的作用。

4.1.1 BERT

2018 年，Google 提出 BERT 预训练（pre-trained）模型[17]，BERT 代表 Bidirectional Encoder Representations from Transformers。BERT 是一个性能卓越的自然语言处理模型，它的出现显著提升了多个自然语言任务的最佳性能。BERT 框架中存在两个重要的步骤：预训练（pre-training）和微调（fine-tuning）。

4.1.1.1 概念介绍

“预训练”是指使用某种策略将大量多样化的自然语言数据集样本预先训练模型，使模型学习数据集中的共性。BERT 预训练模型采用的策略之一是 MLM（Masked language modeling[46]，掩码语言建模），该策略是一种自监督学习方法。在 MLM 训练时，对输入的词元序列随机掩盖部分 token，训练的目标是仅根据上下文对被掩盖的 token 进行预测。MLM 方式同时利用上下文，契合对深度双向变换器（deep bidirectional Transformer）的预训练。此外，BERT 的预训练任务还包括预测下一句话（Next Sentence Prediction），通过输入一对句子 (A, B)，让模型预测 B 是否为 A 的下一句话。在训练数据中，50%的 B 为 A 的真实下一句，50%的 B 为随机采样。该预训练任务对问答系统（Question Answering，QA）和自然语言推理（Natural Language Inference，NLI）等需要理解句子之间关系的任务有重要作用。BERT 使用的预训练数据集包括 BooksCorpus、English Wikipedia 等。

“微调”是指在具体的任务当中，对源模型（预训练模型）使用目标领域数据集进行二次训练，目标是使预训练的模型更适合当前任务。对模型微调的计算资源要求较低，而预训练阶段需要消耗大量的计算资源，因此在实际使用中，多数研究人员采用预先训练好的 BERT 模型，再通过目标数据集进行微调，即能取得不错的性能。本毕业设计同样采用该方法。

4.1.1.2 BERT 的架构

BERT 由三层组成，第一层为输入表示层即 Embedding 模块，将 token 转化为向量，使之能够计算输出。中间层为双向 Transformer 层，Transformer 层通过矩阵乘法、共次加法、状态传递等算法将输入的单词向量转换为向量输出。最后一层为预微调模

块，可以根据具体的任务进行调整。例如将 BERT 用于 baseline 标签方案，得到的模型结构如图 4-1 所示，其中 Transformer 层数与具体采用 BERT 模型有关，在 $BERT_{BASE}$ 中为 12，在 $BERT_{LARGE}$ 为 24。

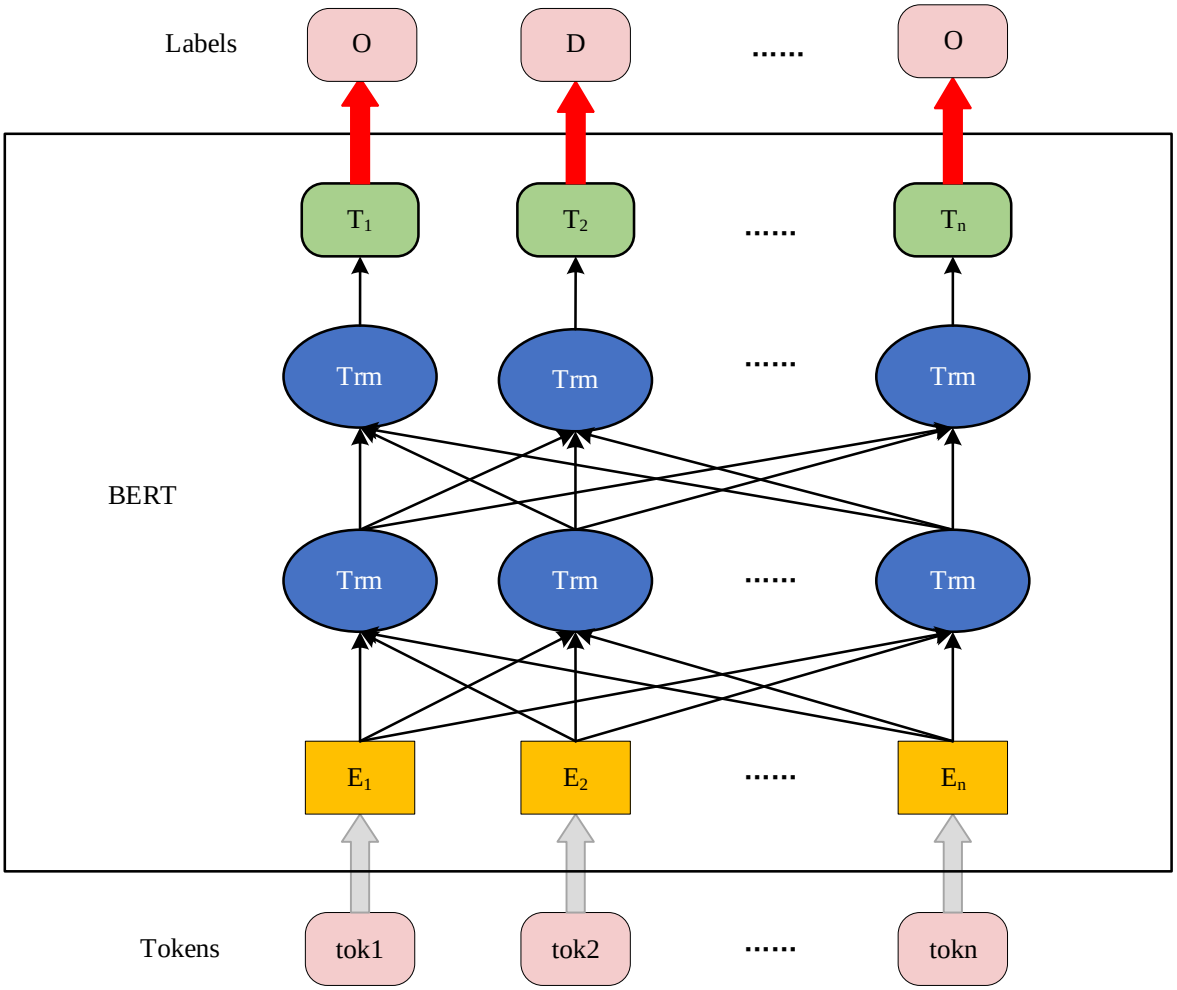


图 4-1 BERT 结构

上图中，E_1 为 Embedding 层中 token1 的输入嵌入表示。BERT 的输入表示层与简单的词向量（Word2Vec）不同，BERT 的输入嵌入表示 E 考虑了上下文，由三种嵌入向量的求和得到的。三个嵌入向量分别为：词嵌入（token embeddings）、文段嵌入（segmentation embeddings）和位置嵌入（position embeddings）。示意图如图 4-2 所示。

词嵌入向量是词汇表中学习到的词嵌入表示，将每个词转化为 768 维的向量。

文段嵌入在句对任务中起到关键作用。在一对句子 (A, B) 中， A 句的文段嵌入表示为 E_A， B 句的文段嵌入表示为 E_B，以此区分两个拼接在一起的句子。

位置嵌入表示是对 Transformer 缺陷的补充，主要是因为 Transformer 与 RNN 不同，无法自动捕捉序列中元素的顺序性。因此，在使用 Transformer 进行序列建模时，需要为每个元素指定其在序列中的位置信息，以便 Transformer 能够区分不同位置的元素。

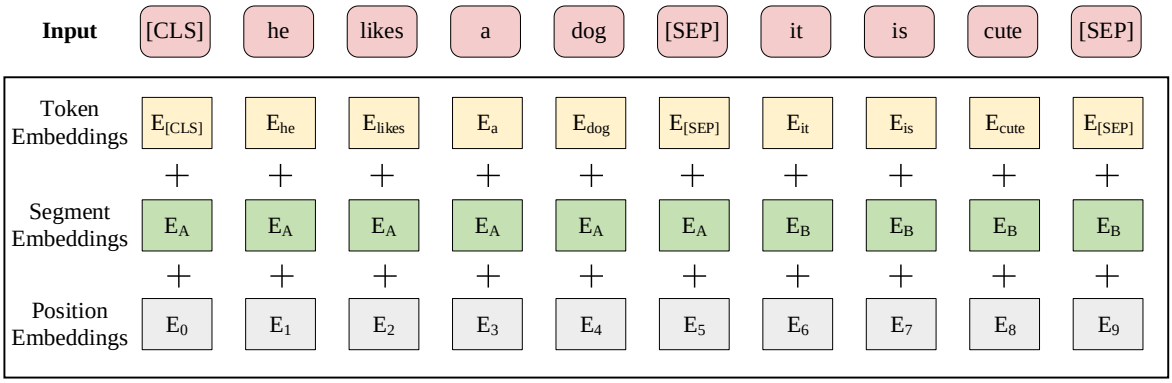


图 4-2 嵌入表示层

4.1.2 BERT-CRF

在上述技巧和大量数据预训练的基础上，BERT 预训练模型可以极大地提升序列标注效果，因此本毕业设计利用 BERT 强大的信息记忆和抽取能力，将 BERT 作为模型的最底层，获取上下文文本信息。

在 BERT 层后增加一个全连接层便于对 token 序列进行标注。标注层可以采用 Softmax 层或 CRF 层，本设计中采用 CRF 层作为标注判断层。

Softmax 层在神经网络中常常作为输出层，可以提高神经网络的学习速度，但是没有考虑上下文 token 之间的信息，缺乏全局视野，CRF 很好地弥补了这一点，以下简单介绍 Softmax 函数。

4.1.1.1 Softmax 层

Softmax 中文名称为柔性最大值。以序列标注模型为例，假设神经网络或是深度学习网络在输出之前得到一个向量矩阵 *Logits*，*Logits* 为一个 $L \times T$ 的二维矩阵，其中 L 为一个句子的长度，T 为标签数量。Softmax 层即对 L 个标签的分布进行归一化，将概率最大的标签作为输出结果。Softmax 函数如式 4-1。

$$a_i^j = \frac{e^{z_i^j}}{\sum_{i=1}^{T} e^{z_i^j}} \tag{4-1}$$

其中，a_i^j 表示第 j 个 token 的第 i 个标签的归一化概率，z_i^j 表示 *Logits* 矩阵中第 j 行第 i 列的值。

根据公式 4-1 定义可知：

$$\sum_{i=1}^{T} a_i^j = 1 \tag{4-2}$$

Softmax 函数得到的值都为正数，且柔性最大值层的输出相加为 1，因此可以将该层的输出视为概率分布，选择概率最大的标签作为预测结果。如图 4-3 所示。

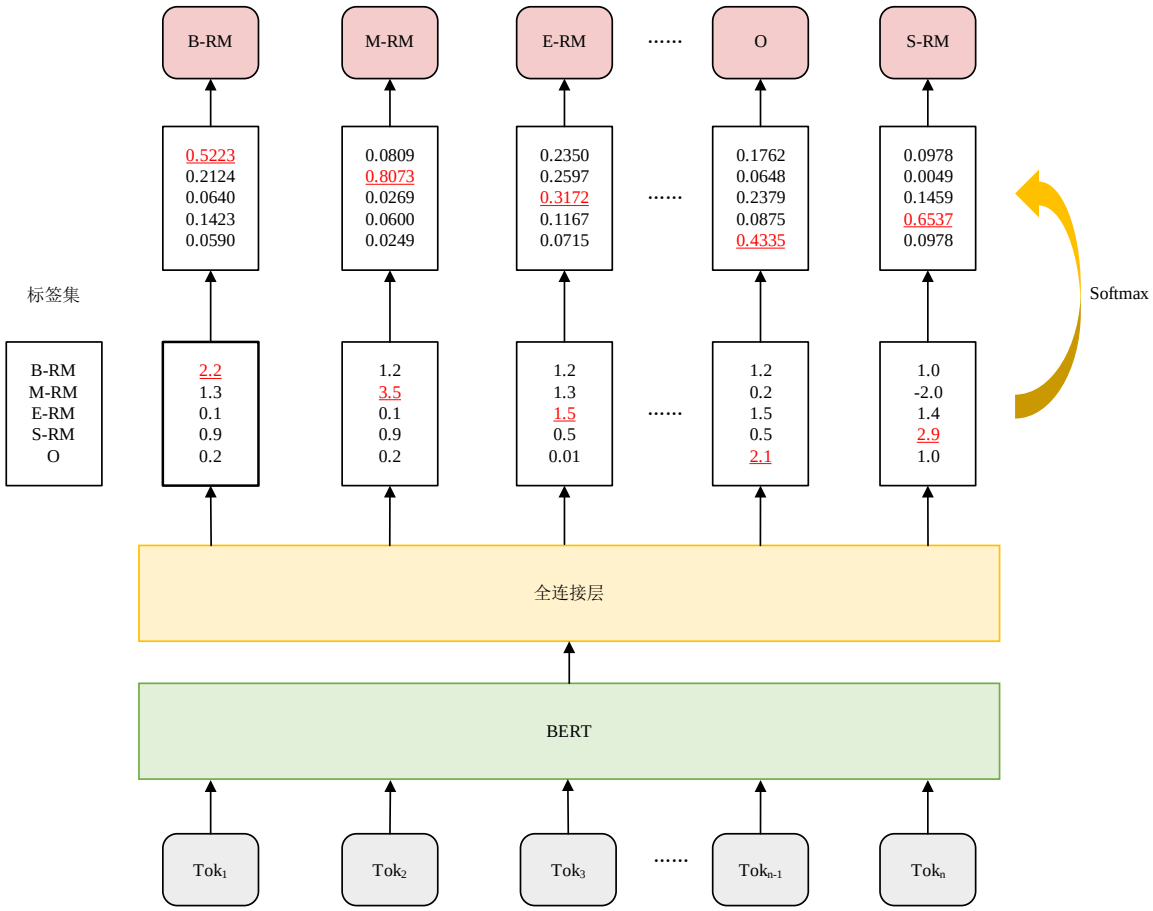


图 4-3 采用 Softmax 层标注示意图

4.1.1.2 CRF 层

由一小节可知，Softmax 预测时同一序列中的 token 之间相互独立，预测结果仅与对应 token 的输出有关，没有相互影响。这在本毕业设计任务当中，是不合理的，因为文本顺滑任务检测不流畅成分需要结合上下文才能判断。故采用 CRF 层作为标签预测层。

CRF 是以标签路径为预测目标，通过在 *Logits* 添加一些约束——转移矩阵，以确保预测的标签的合法性，这在细粒度标签方案中是有必要的（细粒度标签方案中标签之间自带上下文关联），约束会在训练过程中自动学习。转移矩阵亦可在一些非法转移之间进行初始化约束限制，这个过程称为约束解码或限制解码。在本毕业设计中约束解码对模型效果并没有太大影响，但是可以为 Softmax 层提供一种约束思路，如[47]中的研究所述。

CRF 是一种基于图模型的判别式模型，常用于序列标注任务。CRF 的优点之一是可以考虑全局特征，从而实现全局最优标签路径的搜索。为了在 CRF 中实现全局最优标签路径的搜索，通常会使用动态规划算法——维特比解码算法。在 2.2.2 节中，已经详细介绍了维特比解码算法的实现过程和原理。通过使用维特比解码算法，CRF 可以在保证全局最优的情况下，高效地求解标签路径，从而提高模型的精度和效率。

BERT-CRF 模型及 CRF 层图解如图 4-4 所示。

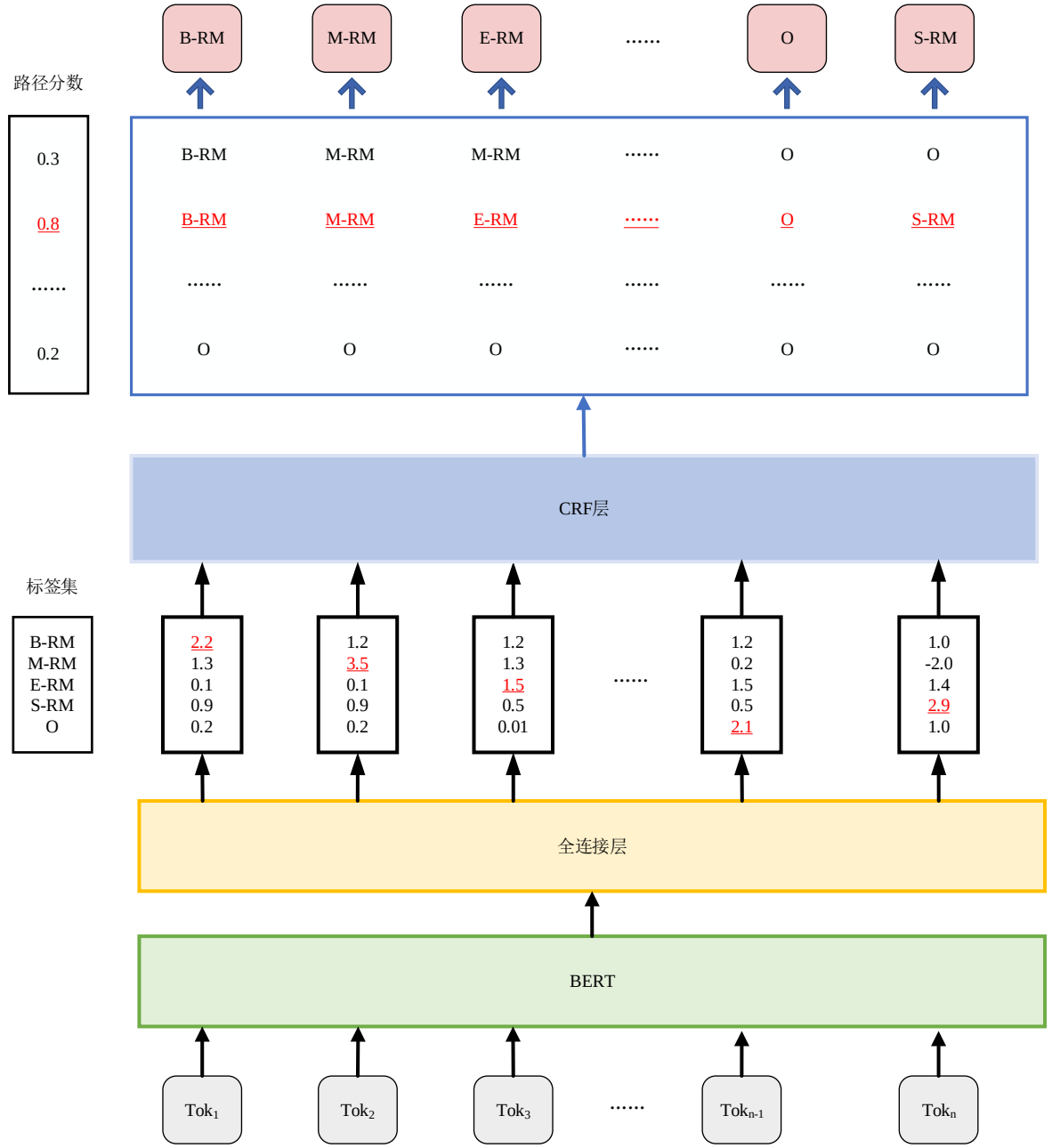


图 4-4 BERT-CRF 模型

4.2 模型建立流程与数据预处理

在本设计中，采用 BERT-CRF 模型作为序列标注模型。具体来说，BERT 层负责抽取文本序列的信息，CRF 层则根据 BERT 层的输出对观测序列进行标注。在 4.1 节中，我们对 BERT-CRF 模型进行了详细的描述和介绍。接下来，4.2.1 和 4.2.2 节将进一步介绍模型建立过程中使用的技术细节。此外，在进行模型训练之前，我们还需要进行数据预处理工作。

4.2.1 模型建立流程

4.2.1.1 环境依赖

本毕业设计中的序列标注模型 BERT-CRF 在 PyTorch 和 Transformer 环境依赖下完成。

PyTorch 一个基于 Torch 库的 Python 深度学习框架，由 Facebook 开源，目前被广泛应用于学术界和工业界。相较于 Tensorflow2.x，API 设计更加简洁、优雅和易懂；PyTorch 调试方便，十分灵活，便于理解代码；PyTorch 还提供完整的文档和高效的指南，以及社区支持。

使用 BERT 或 Transformer 模型都需要使用到 HuggingFace 提供的 Transformer 依赖。HuggingFace 的官方网站为：http://www.huggingface.co，在这里有各类的 BERT 预训练模型（Models）、数据集（Datasets）、文档（Docs）、甚至自然语言处理课程。HuggingFace 中可下载基于 PyTorch 实现的 BERT、BERT 的 TensorFlow 版以及其他主流的预训练模型。

BERT 模型存在众多预训练模型类别，例如使用较多的 bert-based-cased 和 bert-based-uncased，前者与后者的区别主要在于后者不区分英文的大小写。Bert-based-uncased 更为常用，但是在一些特定的领域会要求区分大小写使用 bert-based-cased，比如命名体识别。在文本顺滑任务当中，英文字母的大小写并不重要，且对模型无必要影响，在数据处理过程中会将所有文段全部小写化（lower-case），因此采用 bert-based-uncased 预训练模型。

CRF 层采用开源库 SuPar 中的高度并行化实现方式，主要目的是加快计算速度。SuPar 主要是由苏州大学博士生张宇以及其所在课题组的同学在李正华老师的指导下共同开发的，是一个专为结构化预测而设计的 Python 包，包括许多最先进的句法/语义解析器的再现（具有超过 19 种语言的预训练模型），以及几种著名的结构化预测算法的高度并行化实现。

本毕业设计中为建立模型所安装的环境及使用依赖的版本、BERT 模型以及 CRF 层如表 4-1 所示。

表 4-1 环境依赖

环境依赖或模型类别	版本号或名称
python	3.8.13
torch	1.12.1
transformers	4.25.1
BERT 模型	bert-base-uncased
CRF	SuPar

4.2.1.2 模型搭建

在构建虚拟环境、完成环境依赖的安装后，下载 bert-based-uncased 预训练模型，并根据毕业设计任务简单修改原 SuPar 中的 CRF 层，基本上完成了模型构建所需的准备工作。

在 4.1.2 节中已详细介绍了本设计采用的序列标注模型：BERT-CRF。因此在本节不再赘述，仅给出模型建立和前向传播流程图。

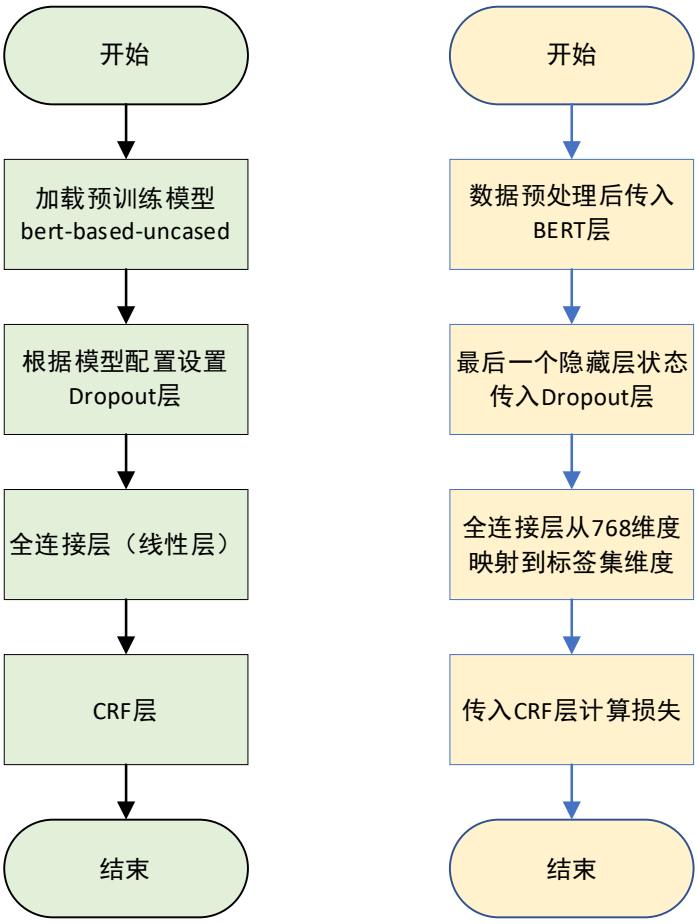


图 4-5 模型构建及前向传播流程图

在模型建立和前向传播的流程图，以及模型结构图中，可以看出序列标注模型的简单易使用性，三者之间存在极大的相似性。

在流程图中提到了 Dropout（弃权）技术。由于 SWBD 数据集样本数量较少，训练结果可能存在过拟合的情况，因此采用弃权技术减缓或防止过拟合的情况。顾名思义，所谓“弃权”是指在前向传播的过程中，使某一个神经元以一定概率“休眠”，即让神经元停止工作，丢弃该神经元的输出。弃权技术可以在一定程度上增强模型的泛化能力，使模型不过分依赖于某些局部特征。但是在模型评估或者预测的情形下，应该屏蔽 Dropout 层，这在 PyTorch 中容易实现。因为模型存在两种模式，train 模式和 eval 模式，训练时采用 train 模式，预测评估采用 evaluate 模式。模型在 eval 模式下会自动屏蔽 Dropout 层，但是在 train 模式不会。

4.2.2 数据预处理

4.2.2.1 对齐问题

数据处理仅得到文本序列与标签序列，但是在模型训练中，需要将文本转换为token ID，也就是序列化，模型才能理解文本。在 BERT 中，并不需要额外构建词典来完成这个操作，直接调用分词器 tokenizer 的 API 即能很好地处理这个问题。此外，标签集也需要序列化，标签集的序列化并不需要额外处理，以数字 $\{0,1,2,...,T-1\}$ 唯一映射每一个标签即可。

在模型结构图中，输入部分引入了“[CLS]”和“[SEP]”这样特殊的 token，4.1.2 节中并没有详细解释。这部分特殊的 token 是由 tokenizer 引入的。BERT 中引入的所有特殊 token 的表 4-2 所示。

表 4-2 特殊 token 含义及编号 ID

特殊 token	含义	ID
[CLS]	出现在句首，用于下游句子分类场景。	101
[SEP]	分隔符，分开两个句子。	102
[PAD]	填充符，将不同长度的句子填充到同一个长度。	0
[UNK]	未登录词，表示词典中不含该词。	100
[MASK]	掩码。在预训练任务中使用。	103

当对分词器 tokenizer 初始化以后，直接将我们得到的文本序列输入分词器 tokenizer 即可以得到 BERT 直接使用的输入。

但是在分词器 tokenizer 使用的过程中，需要解决“对齐”问题。对齐问题的主要原因在于 tokenizer 分词器会将某些单词分解为多个子词。图 4-6 所示即为对齐问题。

原始语料： and [you, + one's] feeling obligated

	文本序列	and	you	,	one’s	feeling	obligated					
	标签序列	O	D	D	O	O	O					
分词后文本序列	[CLS]	and	you	,	one	’	s	feeling	ob	##liga	##ted	[SEP]
对齐后的标签序列	?	O	D	D	O	?	?	O	O	?	?	?

图 4-6 分词后的对齐问题

在上图所示的内容中，仅展示了 baseline 标签方案下，分词后的文本序列与标签序列没有匹配的情况（标签序列仅有 6 个元素，而新的输入有 12 个元素）。其中，问号所示内容则为对齐问题所关注的——在 tokenizer 产生的特殊 token 和分出子词情况下，应该如何对齐标签序列。

关于上述问题，对于特殊 token 的处理较为简单，可以在前向传播和反向传播的过程中忽略该部分，这样在预测和更新参数时特殊的 token 不会发挥作用。具体方式为可暂时将特殊 token 的标签设为-100，在前向传播和反向传播中在 BERT 层和 CRF

之间屏蔽特殊 token 位置的值，使之不参与 CRF 层标签预测即可。

某些词产生的子词对齐方式有两种方案可选。第一种方案类似特殊 token 的处理方式，仅考虑原词细分后的第一个子词，其余子词均不考虑，这是为了避免分解成大量子标记的长词对损失计算造成严重影响。第二种方案则是将子词依据原词标注结果进行匹配，但是需按照原标签方案进行。

本毕业设计采用第二种方案。在 baseline 标签方案中，子词的标注结果与原词一致，仅需词与原词的映射关系即可解决对齐问题，但是在细粒度标签方案中由于前缀标签需要额外讨论。Tokenizer 分词器中有记录该映射关系的方法：word_ids()。

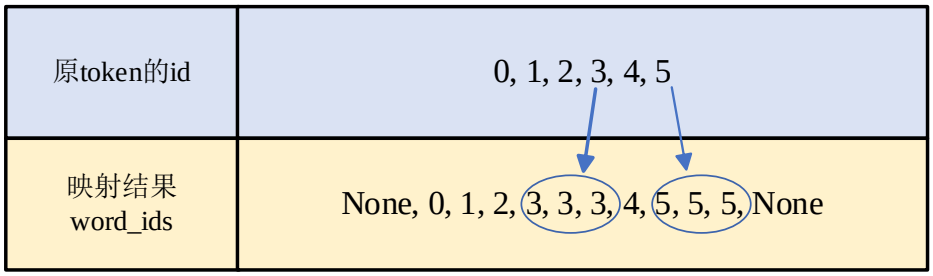


图 4-7 分词前后下标索引映射

图 4-7 为示例结果，其中“[CLS]”和“[SEP]”特殊 token 的映射情况用 None 表示，因为原文本序列中并不存在“[CLS]”和“[SEP]”。

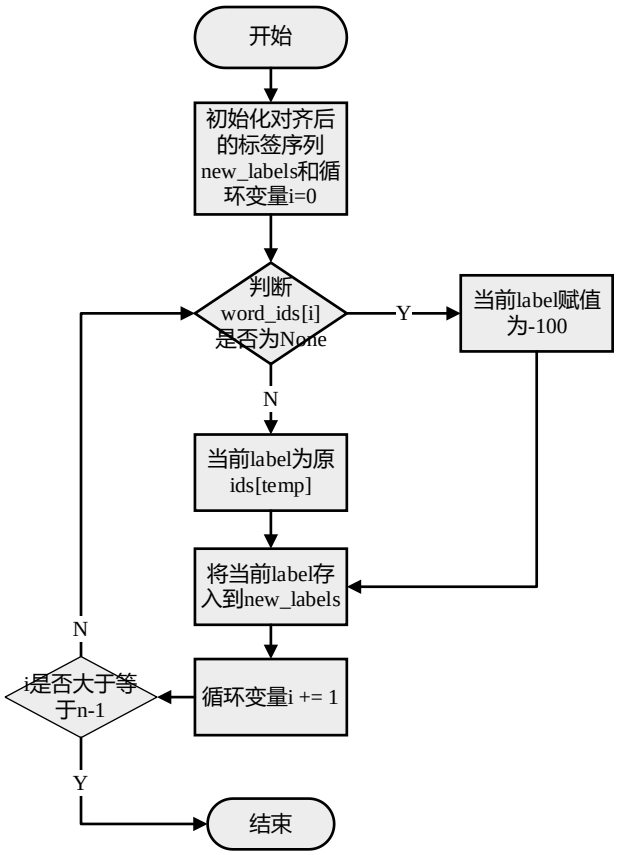


图 4-8 baseline 方案对齐处理流程图

图 4-8 所示为 baseline 标签方案的对齐过程。遍历映射结果序列 word_ids 之前初始化循环变量 i 和目标序列 new_labels，遍历过程中仅需判断当前 token 的 id 值是否为 None，若为 None 则将当前标签置为-100，反之则为原 token 对应的标签，由于 baseline 标签方案无需额外规则，因此上述流程十分简单。

反观细粒度标签方案，则需要一些规则保证新标签序列的合法性。

非嵌套细粒度标签方案的标签集如表 4-3。

表 4-3 非嵌套方案标签集

非嵌套标签方案	O, B-RM, M-RM, E-RM, S-RM, B-IM, M-IM, E-IM, S-IM, B-RP, M-RP, E-RP, S-RP

所有子词出现情况的对应规则如表 4-4。

表 4-4 非对齐规则

规则	含义
规则 1	原词标签前缀为“B-”，若子词为第一个子词，则和原标签一致，反之则前缀修改为“M-”；
规则 2	原词标签前缀为“M-”，则所有子词标签保持不变；
规则 3	原词标签前缀为“E-”，除最后一个子词标签外皆将前缀修改为“M-”，最后一个子词标签与原词标签一致；
规则 4	原词标签前缀为“S-”，第一个子词标签修改为“B-”，最后一个子词标签修改为“E-”，若存在其余子词，其余子词标签修改为“M-”；
规则 5	所有后缀保持不变。

而嵌套细粒度标签方案的对齐方法，对于每一个嵌套标签都需要遍历其内的子标签，子标签依然适用上述规则。在标签对齐的过程中，嵌套标签方案可能会生成原标签序列中本来不存在的嵌套标签，因此需要及时更新标签集，防止模型在训练过程中出现无法识别的标签错误。

4.2.2.2 批处理

至此，已详细描述了数据预处理中最为复杂的工作，还有一个称为训练数据批处理（batching）的操作没有介绍，由于较为简单且实现方便，以下仅作简单介绍。

训练数据批处理有便于并行计算，减少内存开销，加快计算速度，增强模型的稳定性，减弱过拟合等作用，。在自然语言处理中，大多数句子的长度是不相等的，无法直接存入一个等长的批次（batch）当中，因此需要将其填充至其中最长序列的长度，填充值一般设置为 0。

在本设计当中，这个操作同样可由分词器 tokenizer 完成，仅需将分词器 tokenizer 的参数 padding 设为 True 即可。

4.2.2.3 滑动窗口处理

经过上述众多数据处理与数据预处理操作，已为模型训练做好了准备，可开展对序列标注模型训练的工作。本小节简要介绍本毕业设计中使用的滑动窗口技术，由于该操作即使舍去也不会影响模型的训练，因此置于最后一节讲述。

在 BERT 模型的 tokenizer 分词器中，模型参数中存在“最大序列长度（max_length）”和“截断（truncation）”两个可选参数。最大序列长度参数的含义为模型可接受的最大序列长度，可设置为给定整数；截断参数的含义为当序列长度超过最大长度时是否将其截断，是否丢弃超过最大长度的值，可设置为布尔值，默认为真。

假设设置了最大长度 *len*，且将截断参数 truncation 设置为真。当语料集中的文本序列长度超过最大长度 *len* 时，则在处理时则会丢弃大于最大长度的部分。从充分利用语料集出发，这样的操作有点过于“粗暴”，可能损失了包含语义价值的信息。为避免上述情况，有许多解决方案，其中一种解决方案即是滑动窗口技术。除滑动窗口方法外，最简单的方法为将最大长度参数设置为语料集中出现的序列的最大长度，但是该方法会将较短的序列填充大量的“[PAD]”标签，导致占用过多显存，降低模型训练速度和推理速度。

对于过长的序列，滑动窗口技术以一定大小的窗口对该序列进行“滚动”变化，将原序列变为多个长度小于等于最大长度的序列。流程图如 4-9 所示。

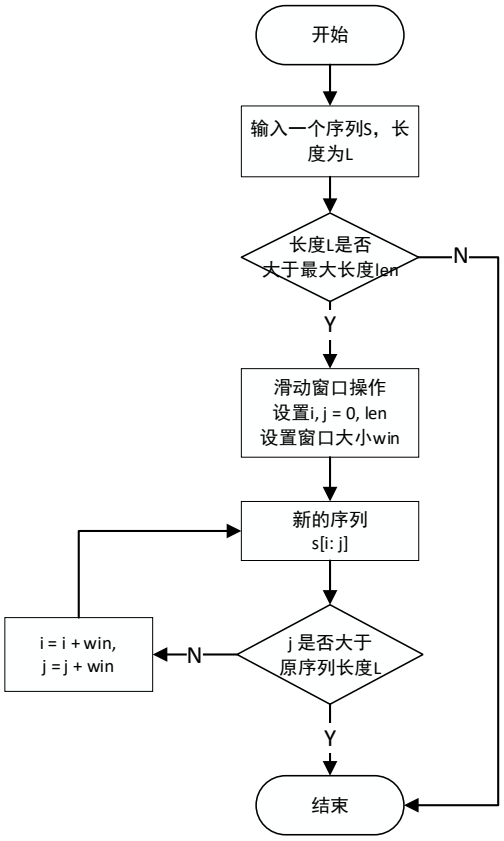


图 4-9 滑动窗口操作流程图

在滑动窗口操作中，窗口大小 win 一般设置为 5。图 4-9 流程简要叙述如下：输入一个长度为 L 的文本序列 S，如果长度 L 大于最大长度 len，则进行滑动窗口处理。初始化两个下标 $i=0$ 与 $j=len$，i 记录序列开始部分，j 记录序列末尾部分。完成初始化后，原序列得到的第一个子序列为 0 到最大长度 len 的序列 $S[i:j]$。随后判断 j 是否大于序列长度 L，若否，更新 i 与 j，每次更新步长为窗口大小 win，得到第二个子序列 $S[i:j]$，重复该过程直到 j 大于长度 L 退出循环。

滑动窗口操作结束后，一个大于最大长度的序列可得到多个小于等于最大长度的子序列，经滑动窗口操作可以使模型充分利用语料集中的信息，且在训练过程中有效降低显存占用，提高模型的训练速度。

4.3 本章小结

本章第一节详细描述了所采用的 BERT-CRF 序列标注模型结构，介绍了 BERT 的预训练任务、微调、结构、和 embedding 层等相关知识，并且对比了 CRF 与 Softmax 层——CRF 层能提供“全局视野”，得到全局最优解，与文本顺滑任务需上下文检测不流畅成分相契合，这是 Softmax 所不具备的。

本章第二节简要讲述了模型建立所需的环境依赖以及流程，着重对数据预处理工作中不同标签方案的“对齐”操作进行了详细的描述，为后续模型训练工作做好了准备。

第 5 章 实验与分析

5.1 模型训练与评估

通过目前已经得到的可直接使用的数据集对 BERT-CRF 模型进行参数优化的过程称为模型训练。模型训练的目的是使得模型能够更好地预测新的未知数据。

根据训练好的模型进行性能评估的过程称为模型评估。模型评估的目的是检查模型的泛化能力，能否在未见过的数据集上表现良好。

SWBD 数据集划分为 4 个集合：训练集、开发集（验证集）、测试集和留作未来使用的部分[48]。本论文不使用最后一个集合的数据。

表 5-1 语料集划分

集合	SWBD 文件范围
训练集	文件夹 2 和 3 中的所有语料
开发集	文件夹 4 中 SW4519 到 SWBD4936
测试集	文件夹 4 中 SW4004 到 SWBD4153
留作未来使用	文件夹 4 中 SW4154 到 SWBD4483

训练集是用来训练模型的数据集，模型在训练集上进行参数优化，以提高模型在训练集上的性能。

开发集（也称为验证集）是用来评估模型性能和调整模型超参数的数据集。开发集数据不参与模型的训练，而是用来评估模型在未训练过的数据上的性能表现，并通过调整超参数来改善模型的性能。

测试集是用来评估模型在“未见过”的数据上性能的数据集。测试集数据不参与模型的训练和调整，只是用来测试模型的泛化能力和性能。测试集是与训练集和开发集互斥的，即测试集中的数据不应该出现在训练集和开发集中。

模型训练和评估过程中，我们首先使用训练集训练模型，然后使用开发集评估模型性能和调整超参数，最后使用测试集来评估模型的泛化性能和性能表现。训练集、开发集和测试集的划分关系是固定不变的，以保证模型的泛化能力和性能可靠性。

5.1.1 模型训练细节

模型训练过程中超参数设置与调整是十分重要的。在本毕业设计中，超参数主要包括：学习率、正则化参数、数据批次大小、迭代次数、Dropout 概率、CRF 层参数初始化、优化算法。

在 BERT-CRF 模型中，学习率的设置在 BERT 层和 CRF 层是不一致的[49]。BERT 在大量语料的预训练后，对语言的拟合效果已经很充分，在微调阶段仅需较小的学习

率即可使模型较快收敛。CRF 层则需从头训练转移矩阵，因此需要更大的学习率加快训练速度使模型较快收敛以达到和 BERT 层相近的收敛速度。在本毕业设计中，将 CRF 层的学习率设置为 BERT 层学习率的十倍。下图所示为 CRF 层学习率 10 倍或 20 倍于 BERT 层学习率的训练情况（删除标点的数据在开发集上的表现）。

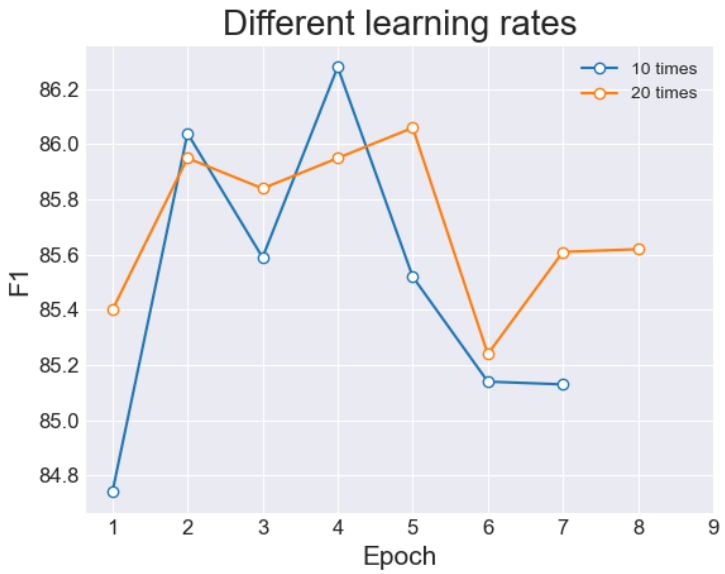


图 5-1 CRF 层不同学习率训练表现

学习率、正则化参数和 Dropout 层的设置都参考了 BERT 论文[17]在的设置。批次大小限于 GPU 设备仅能采用 32，采用 64 的大小会导致显存不够的问题无法训练。迭代次数设置为 15 在 BERT 微调中为一个较大的数，训练过程中在 3 到 5 次迭代模型即能收敛。允许不增长迭代次数是指模型收敛到当前最佳性能后，保存参数后可以继续训练的迭代次数，若在该迭代次数内产生新的最佳性能，可重复该过程，直到不能训练出新的最佳性能或模型训练超过最大迭代次数。

本毕业设计经过调参后采取的最佳参数如表 5-2 所示。

表 5-2 参数设置

超参数	设置
学习率——BERT 层	2e-5
学习率——CRF 层	2e-4
正则化参数——weight decay	0.01
数据批次大小——batch size	32
最大长度——max length	75
迭代次数——epoch	15
允许不增长迭代次数——no-growth epoch	3
Dropout 层概率——p	0.1
优化算法	AdamW

5.1.2 模型评估方式

在模型训练和测试过程中，评估方式是不可或缺的。在序列标注任务中，模型的评估主要是依据 2.3 节中所介绍的三个评估指标：精确率、召回率和 F1 值。在文本顺滑任务当中不考虑准确率指标，因为存在极端情况：倘若将所有文本均视为不顺滑，则准确率为 100%，但是这样的模型并没有意义。

本毕业设计中的不同标签方案中存在一个值得探讨的问题，即如何衡量何种标签方案的数据集训练出的模型效果更佳、泛化能力更强。在文本顺滑任务当中，正如前文所提到的，大多数研究者不关注顺滑部分词，只关注不顺滑部分词。因此在模型评估的过程中，也需遵从类似的思想。模型评估并不以整体顺滑与非顺滑语料的预测情况为目标，而是只关注不顺滑词的识别情况，即不以整体 F1 分数等信息作为衡量标准，而是只以不顺滑词的 F1 分数作为模型性能判定的准则。

根据上述内容，在不同的标签方案中，baseline 标签方案则只需关注 D 标签的 F1 值即可，而在细粒度标签中采用同样的思想，只关注 RM 标签。但是，细粒度标签方案中可能存在一个疑惑。例如，一个词元 token 实际标签为“B-RM”，但是模型将其识别为“S-RM”，本论文依旧将这种情况归于识别正确用于计算 F1 分数。上述解决方法的原因在于：在文本顺滑模型实际应用中，当模型最终判断一个词元 token 属于“RM”标签类时，无论是前缀如何，都会将该词元 token 从原始文本中删除。因此，在评估时考虑标签前缀并无太大意义。如图 5-2 所示。

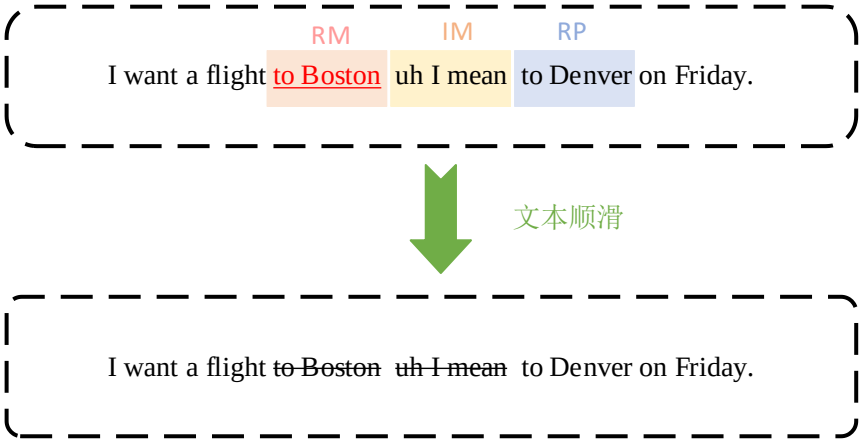


图 5-2 文本顺滑全处理示意图

在图 5-2 中，可以看出 IM 部分在得到最终顺滑文本时，依然需要删除，但是 IM 部分识别简单并不在文本顺滑任务研究范围。

综上所述，在评估细粒度标签方案及 baseline 标签方案训练得到的模型时，为保持评估方法的一致性与合理性，将细粒度标签方案中任意包含“RM”的标签均视为与 baseline 方案中等价的“D”标签，而不讨论“RM”标签的细分评估结果。在这基础上计算精确率 P、召回率 R 和 F1 分数。

5.2 实验及结果分析

5.2.1 训练结果

在数据处理过程中，存在两种数据：保留了原语料中的标点符号和删除了原语料中的标点符号的数据。在以下大部分讨论的中，以保留了标点符号的数据为主，使用删除标点符号的数据的图像会额外说明。

在实验过程中，由于细粒度标签方案的可塑性，因此本设计采用的细粒度标签方案并不止一种，具体情况如表 5-3 所示。

表 5-3 本设计中所有细粒度标签方案

细粒度标签方案	标签集大小	标签集	训练的模型编号
考虑嵌套情况	248	O, B-RM, M-RM, E-RM, S-RM, B-IM, M-IM, E-IM, S-IM, B-RP, M-RP, E-RP, S-RP, <PAD>, B-RM|B-RM, M-RM|M-RM, M-RM|E-RM……	模型 1
考虑嵌套 仅 RM 有前缀	146	O, B-RM, M-RM, E-RM, S-RM, IM, RP, <PAD>, B-RM|B-RM, M-RM|M-RM, M-RM|E-RM……	模型 2
考虑嵌套 且无前缀	59	O, RM, IM, RP, <PAD>, RM|RM, RM|IM……	模型 3
忽略嵌套情况	14	O, B-RM, M-RM, E-RM, S-RM, B-IM, M-IM, E-IM, S-IM, B-RP, M-RP, E-RP, S-RP, <PAD>	模型 4
忽略嵌套 仅 RM 有前缀	8	O, B-RM, M-RM, E-RM, S-RM, IM, RP, <PAD>	模型 5
忽略嵌套 且无前缀	5	O, RM, IM, RP, <PAD>	模型 6

在上表中，细粒度标签方案各不相同，但是上表并不是所有可能标签集情况，还可根据不同的标准设计不同的方案，本毕业设计限于时间、实验设备等原因，仅讨论以上列举的标签集训练结果。

模型训练过程是一个迭代过程，其目标是最小化损失并调整模型参数。在训练过程中，使用训练集来计算损失值，并使用优化算法来更新模型参数，以使损失值最小化。随着模型参数的调整，训练集的损失值逐渐减小，如图 5-3 所示。图 5-3 中展示了 baseline 标签方案和非嵌套细粒度标签方案（模型 4）的训练损失变化情况。从图 5-3 可以看出，模型 4 和 baseline 模型训练过程中，模型收敛速度基本相当。这说明，与 baseline 标签方案相比，非嵌套细粒度标签方案在训练过程中没有明显的差异，因此认为细粒度标签方案是可行的。

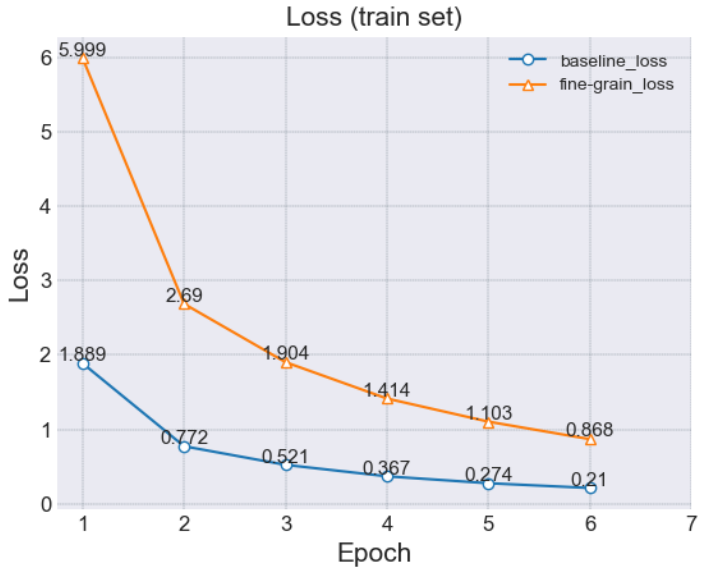


图 5-3 迭代训练时损失变化曲线

在模型训练过程中，即在损失 loss 不断减小的过程中，以模型在开发集上的性能表现为依据，选择表现最佳的模型进行保存。迭代过程中 baseline 标签方案和非嵌套细粒度标签方案对应的模型在开发集性能变化如图 5-4 所示。

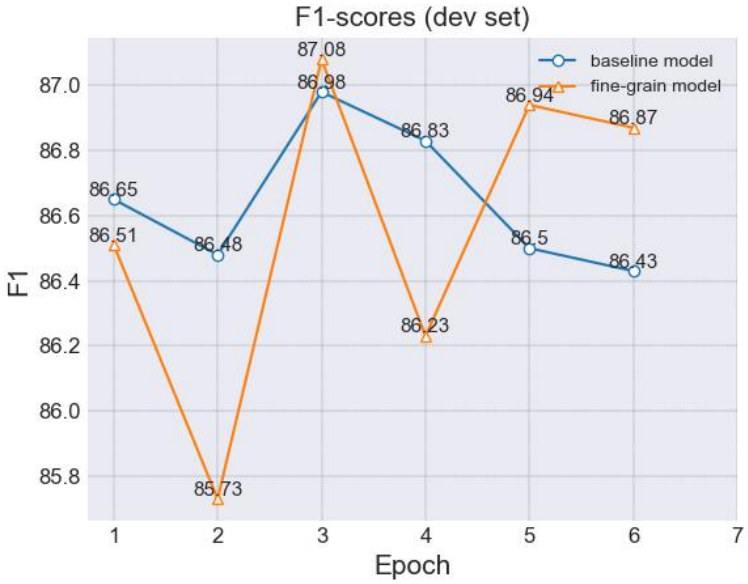


图 5-4 迭代训练时模型在开发集的表现

从图 5-4 可以看出，非嵌套细粒度标签方案（模型 4）和 baseline 标签方案相比，在第 3 次迭代均达到最佳性能，但是前者表现超过了后者，这样的结果说明细粒度标签方案在一定程度上提升了模型的性能，不过还需在测试集结果上进一步实验。

在开发集上，baseline 标签方案和各个细粒度标签方案的最佳训练结果 F1 值性能如图 5-5 所示。

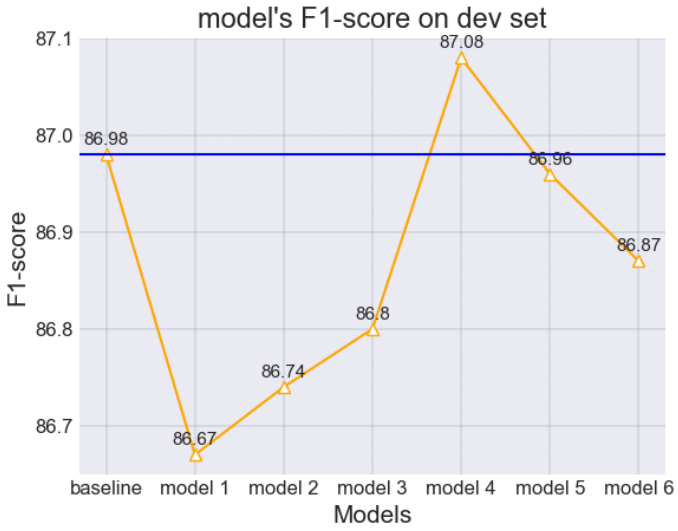


图 5-5 各个模型在开发集的最佳表现

上述在开发集上性能最佳的模型，需要在测试集上检验模型的泛化能力和性能可靠性。各个模型在测试集的表现如图 5-6 所示。



图 5-6 各个模型在测试集的表现

由图 5-6 可知，模型 4 在测试集上的性能表现最佳，此外除模型 5 外的细粒度标签方案都在测试集上的表现超越了标签方案。说明在细粒度标签方案的训练数据中，模型能学到 baseline 标签数据所不具备的更多有用信息，尤其是来自标准注释结构的信息或特征，也验证本毕业设计是有益尝试。

5.2.2 实验结果分析

经过 5.2.1 节关于模型训练结果和测试集表现的陈述，认为实验结果达到了本毕业设计最初的目标——探讨细粒度标签方案对模型性能的影响。根据 baseline 标签方案和细粒度标签方案的实验结果相比，序列标注模型 BERT-CRF 在细粒度标签的训练数据上表现更佳。

5.2.2.1 模型性能分析

在这一小节中，选择细粒度标签方案中性能最佳的标签方案——非嵌套且有前缀的标签方案与 baseline 标签方案训练得到的两个模型进行讨论，简称为 fine-grained 模型和 baseline 模型。

在测试集上，baseline 模型和 fine-grained 模型的性能表现具体如表 5-4 所示：

表 5-4 baseline 模型和 fine-grained 模型性能对比

模型	精确率 P	召回率 R	F1 分数
baseline	87.90	84.50	86.16
fine-grained	86.78	87.32	87.05

Fine-grained 模型相比于 baseline 模型在召回率上有所提高，在精确率存在一定下降。召回率的提高说明细粒度模型能够更准确地识别不流畅内容，减少了漏判的情况，提高了模型的可靠性和准确性；而精确率的下降则表示模型在识别不顺滑内容时可能会误判顺滑内容，导致误差增加。综合来看，模型在整体上表现更好，因此 fine-grained 模型综合指标 F1 值相比 baseline 模型提高了 0.89。

上述模型性能对比分析过程中，开发集和测试集中均使用 4.2.2.3 节中提到的滑动窗口技术。在 4.2.2.3 节中，详细描述了该技术的方法流程和使用带来的优点，但是该方法的缺陷也不应该忽略。其中较为明显的缺陷是在训练过程中经滑动窗口“分裂”得到的新序列可能存在文本序列和标注结构不“健全”或难以理解的情况，这种数据对 baseline 标签方案和细粒度方案都存在影响，会使模型学习或评估标签方案不完整的数据。在训练过程中，可以将这样的数据视为噪音或扰动，有助于加强模型的抗干扰能力。但是，在评估过程中可以不采用滑动窗口技术，在更为规范的数据上对模型进行测评，或是可理解为换个角度评估模型的性能。

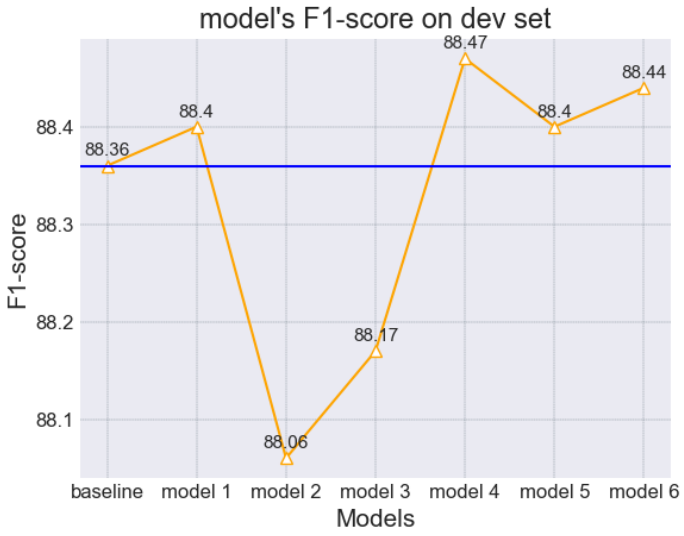


图 5-7 各个模型在不使用滑动窗口处理的开发集上表现



图 5-8 各个模型在不使用滑动窗口处理的测试集上表现

从图 5-7 和图 5-8，即使在不使用滑动窗口处理的数据集上，无论是开发集还是测试集，细粒度标签方案对应的模型（以模型 4 为代表）的性能表现依旧优于 baseline 模型表现。因此，这也从另一个角度说明细粒度标签方案的提出是有效的，模型通过细粒度标签方案标注的数据相比于 baseline 标签方案的确能够在训练过程中学习到更多的特征。

虽然在不使用滑动窗口处理的评估方式下，各模型整体上性能表现更佳，但是为了数据集处理方式统一的严谨性，图 5-7 和图 5-8 所示的评估表现并不能作为模型表现记录，仅仅作为另一视角看代当前模型，为模型分析提供多样的角度。本论文仍以表 5-4 所示的结果作为 baseline 模型和 fine-grained 模型的性能结果。

5.2.2.2 实际语料分析

在 3.1.3.2 中详细介绍过不流畅类型的可以根据标准注释结构中 RP（repair）情况分为三种类型：重启、重复和修正。根据这三种类型可以从实际语料分析 5.2.2.1 中 fine-grained 模型和 baseline 模型的具体表现。

在测试集中，三种类型的分布情况如表 5-5 所示。

表 5-5 测试集中各不流畅类型统计结果

测试集	数量
重启（restart）	265
重复（repetition）	193
修正（correction）	2067

表格 5-5 中，数量是以一个标准注释结构为计量单位，即一个标准注释结构只可分为重启、重复和修正三种类型，但是一个句子不止一个标准注释结构，则一个句子可能包括多种不流畅类型，也可能一种也没有（说明这个句子流畅）。

Baseline 模型在测试集上关于三种不流畅类型的预测结果如表 5-6 所示。

表 5-6 baseline 模型预测情况

baseline 模型	数量	占全部结果比例
重启（restart）	153	57.74%
重复（repetition）	166	86.01%
修正（correction）	1765	85.39%

Fine-grained 模型在测试集上关于不流畅类型的预测结果如表 5-7 所示。

表 5-7 fine-grained 模型预测情况

fine-grained 模型	数量	占全部结果比例
重启（restart）	158	59.62%
重复（repetition）	169	87.56%
修正（correction）	1795	86.84%

根据表 5-6 和表 5-7 相比，fine-grained 模型明显在三种类型的预测结果都略优于 baseline 模型，预测正确的占比都有稳定提升，说明 fine-grained 模型在细粒度标签方案的帮助下能学习到更多与注释结构相关的信息，例如图 5-9 所示从测试集中抽取的一个预测语料结果。

语料　sure, yeah, [I understand, + I understand] your point.

语句	sure	,	yeah	,	i	understand	,	i	understand	your	point	.
Baseline模型	O	O	O	O	O	D	D	O	O	O	O	O
Fine-grained 模型	O	O	O	O	B-RM	M-RM	E-RM	B-RP	E-RP	O	O	O

图 5-9 baseline 模型和 fine-grained 模型预测差异

图 5-7 中的语料为一个较为简单的注释结构，分类为重复不流畅类型。在 baseline 模型实际预测中对不流畅部分（“I understand,”）没有完全预测正确，而 fine-grained 模型在学习了更多注释结构信息的情况下，对该重复类型更为敏感。训练集中的该语料较为简单却扼要地说明注释结构信息对模型预测的帮助。

上列 baseline 模型和 fine-grained 模型两表中的表现结果均只包含在三种不顺滑类型统计之内模型预测正确的情形。实际上，模型做出的不流畅判断会更多。例如在 fine-grained 模型中，全部不流畅类型预测情况如表 5-8 所示。

表 5-8 fine-grained 模型全部预测情况

fine-grained 模型	数量
重启（restart）	510
重复（repetition）	152
修正（correction）	2041

5.2.2.3 存在的问题

在细粒度标签方案当中，非嵌套且带有全部前缀的标签方案训练得到的模型（对应于模型 5）在测试集上性能最佳，而带有大量嵌套标签方案的模型（模型 1、模型 2 和模型 3）的性能明显较差，并且在开发集的表现都不如 baseline 模型。主要原因在于：嵌套标签在语料集中存在较少，而嵌套标签占对应标签集的数量却较大，导致在训练过程中，模型难以从少量的数据中学习到复杂的标签映射。

嵌套结构的数量如表 5-9 所示。

表 5-9 含有嵌套结构的句子在各个集合占有量

集合	嵌套句数量/总句子数
训练集	4035 / 111675 = 3.6131%
开发集	357 / 7456 = 4.7880%
测试集	175 / 4051 = 4.3199%
总语料集	4567 / 123182 = 3.7075%

在整个语料集中，含有嵌套标签的句子约占 3.71%，而在考虑嵌套且带有前缀的标签方案中，嵌套标签有 234 个，占标签集的 94.35%。这样矛盾的数量占比情况导致模型很难从较少的嵌套标签数据中学习到大量的嵌套标签标注方法。这是本毕业设计难以解决的问题，希望在未来的研究中能找到该问题的解决方案。

即使上述问题难以解决，但是在本毕业设计中尝试减少标签集大小的方法缓解该问题，例如模型 2 和模型 3 对应的标签方案——合并部分前缀来减少标签数量。这样的方法是有效的，模型 2 和模型 3 的性能相较于模型 1 都有提升。

5.3 本章小结

本章第一节详细介绍了模型训练过程中的超参数设置，如学习率、正则化参数、迭代次数、数据批次大小等。此外，第一节还介绍了模型评估中关注的是非顺滑部分的 F1 分数，并且对细粒度标签方案可能出现的情况进行了说明。

本章第二节着重展示模型的训练过程和对实验结果的分析，根据得到的最佳模型在测试集上句子的表现探讨模型提升的方面，实验结果表明本次毕业设计标签方案相较于 baseline 标签方案的确有所提升。

结 论

本论文的研究起点在于，针对文本顺滑任务普遍采用基线标签方案标注数据的现状，本论文结合 SWBD 语料使用的标准注释结构，设计了更加细粒度的标签方案，以期通过标签方案的细粒度化来提高序列标注模型的性能。

本毕业设计的核心工作在于以下几点：

（1）将过往研究的标注方案作为基线标签方案，根据 SWBD 语料中的标注结构设计多种细粒度标签方案；

（2）根据基线标签集和细粒度标签集的特点对 SWBD 语料集进行处理；

（3）构建了序列标注模型 BERT-CRF，并完成了数据预处理操作。随后，使用不同标签方案的数据分别训练了模型，并将训练好的模型保存。

（4）在序列标注模型上完成了多种标签方案的实验，并对比了它们的实验结果。根据实验结果，得出如下结论：相较于基线标签方案，细粒度标签方案为 BERT-CRF 模型提供了更多的结构化信息。在相同的评估方式下，使用细粒度标签方案训练的模型表现出更加优越的性能。

（5）此外，本论文还尝试探讨细粒度标签集大小与模型性能之间的关系：在缺乏足够数据的情况下，序列标注模型的标签集过大模型训练结果往往不尽人意，通过合并一定数量标签减小标签集能缓解这种情况。

虽然本论文所做的工作达到了预期的实验结果，但是还存在需完善的地方：

（1）不能很好地解决嵌套注释结构的数据稀疏问题；

（2）受限于 SWBD 数据集规模，模型的泛化性能较差，未来或可尝试数据增强的方式突破语料集规模的限制；

（3）本设计与目前最佳的模型性能存在一定差距，后续应继续尝试提升模型的性能。

致 谢

时光匆匆如流水，不知不觉大学四年已步入尾声。这四年里或期待、或迷茫、或彷徨、或踌躇满志、或心灰意冷……其中每一次经历、每一次情绪的翻涌都是我成长路上珍贵的记忆。

首先，感激母校——西南交通大学对我的培养，大学四年是我人生中知识和阅历增长最快的时光，感谢交大提供的平台让我有机会丰富知识、了解世界的同时保研到苏州大学。

在本毕业设计及论文的工作期间，非常感谢我的硕士导师——李正华老师的悉心指导，李老师对科研的一丝不苟的态度深深启发了我，并且李老师时常教导我们做科研要耐心细心切莫着急，认真琢磨遇到的问题，让我偶尔挫折烦躁的情绪得以抚平。毕业设计期间张磊师兄、马溪师姐还有蒋浩辰师兄为我解答了许多疑惑，在此对他们的耐心帮助表示感谢。虽然我还没有真正见过李老师和各位师兄师姐，却已受益良多，并且感受到课题组和谐的氛围。此外，对负责毕业设计指导的吕彪老师同样表示由衷感谢，吕彪老师在毕业设计工作方面丰富的经验让我少走了许多弯路。

生活中，室友刘奎、任宸睿、张栩宁给我带来了数不清的欢乐，很感谢大学四年有他们的陪伴。我想我永远也不会忘记 5322 寝室。很幸运，我还相识了许多朋友：执着真诚的陈香润，善良耐心的胡静微，憨厚诚恳的张琦。感谢与这么多朋友的相逢为我的大学生活带来了无数快乐的时光。

最后，对我的家人们表示深切的感激。感谢我的父母和姐姐长久以来对我任何决定的理解与支持。

我由衷地感谢在生命中与每一个人的相遇。

参考文献

[1] 方明峰. 中文口语转录中标点预测与不流畅检测研究[D]. 安徽大学, 2020.

[2] He Z, Wang Z, Wei W, et al. A survey on recent advances in sequence labeling from deep learning models[J]. arXiv preprint arXiv:2011.06727, 2020.

[3] 刘浏, 王东波. 命名实体识别研究综述[J]. 情报学报, 2018, 37(3): 329-340.

[4] Li J, Sun A, Han J, et al. A survey on deep learning for named entity recognition[J]. IEEE Transactions on Knowledge and Data Engineering, 2020, 34(1): 50-70.

[5] 李国栋. 文本数据序列标注问题的研究[D]. 哈尔滨理工大学, 2020.

[6] Nadeau D, Sekine S. A survey of named entity recognition and classification[J]. Lingvisticae Investigationes, 2007, 30(1): 3-26.

[7] Sharnagat R. Named entity recognition: A literature survey[J]. Center For Indian Language Technology, 2014: 1-27.

[8] Toutanova K, Klein D, Manning C D, et al. Feature-rich part-of-speech tagging with a cyclic dependency network[C]//Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics. 2003: 252-259.

[9] Hearst M A, Dumais S T, Osuna E, et al. Support vector machines[J]. IEEE Intelligent Systems and their applications, 1998, 13(4): 18-28.

[10] Eddy S R. Hidden markov models[J]. Current opinion in structural biology, 1996, 6(3): 361-365.

[11] Ratnaparkhi A. A simple introduction to maximum entropy models for natural language processing[J]. IRCS Technical Reports Series, 1997: 81.

[12] Lafferty J, McCallum A, Pereira F C N. Conditional random fields: Probabilistic models for segmenting and labeling sequence data[J]. 2001.

[13] 郭丽丽, 丁世飞. 深度学习研究进展[J]. 计算机科学, 2015, 42(5): 28-33.

[14] Huang Z, Xu W, Yu K. Bidirectional LSTM-CRF models for sequence tagging[J]. arXiv preprint arXiv: 1508.01991, 2015.

[15] Yang Z, Salakhutdinov R, Cohen W. Multi-task cross-lingual sequence tagging from scratch[J]. arXiv preprint arXiv: 1603.06270, 2016.

[16] Ma X, Hovy E. End-to-end sequence labeling via bi-directional lstm-cnns-crf[J]. arXiv preprint arXiv: 1603.01354, 2016.

[17] Kenton J D M W C, Toutanova L K. BERT: Pre-training of Deep Bidirectional

Transformers for Language Understanding[C]//Proceedings of NAACL-HLT. 2019: 4171-4186.

[18] Shriberg E E. Preliminaries to a theory of speech disfluencies[D]. University of California, Berkeley, 1994.

[19] Meteer M W, Taylor A A, MacIntyre R, et al. Dysfluency annotation stylebook for the switchboard corpus[M]. Philadelphia, PA: University of Pennsylvania, 1995.

[20] Qian X, Liu Y. Disfluency detection using multi-step stacked learning[C]. Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 2013: 820-825.

[21] Ferguson J, Durrett G, Klein D. Disfluency detection with a semi-markov model and prosodic features[C]//Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 2015: 257-262.

[22] Wu S, Zhang D, Zhou M, et al. Efficient disfluency detection with transition-based parsing[C]//Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). 2015: 495-503.

[23] Wang S, Che W, Liu Y, et al. Enhancing Neural Disfluency Detection with Hand-Crafted Features[C]//Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data: 15th China National Conference, CCL 2016, and 4th International Symposium, NLP-NABD 2016, Yantai, China, October 15-16, 2016, Proceedings 15. Springer International Publishing, 2016: 336-347.

[24] Bach N, Huang F. Noisy BiLSTM-Based Models for Disfluency Detection[C]//INTERSPEECH. 2019: 4230-4234.

[25] Wang F, Chen W, Yang Z, et al. Semi-supervised disfluency detection[C]//Proceedings of the 27th International Conference on Computational Linguistics. 2018: 3529-3538.

[26] Wang S, Che W, Liu Q, et al. Multi-task self-supervised learning for disfluency detection[C]//Proceedings of the AAAI Conference on Artificial Intelligence. 2020, 34(05): 9193-9200.

[27] Wang S, Wang Z, Che W, et al. Combining Self-Training and Self-Supervised Learning for Unsupervised Disfluency Detection[C]//Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP). 2020: 1813-1822.

[28] Wang Z, Wang Y, Wang S, et al. Adaptive Unsupervised Self-training for Disfluency

Detection[C]//Proceedings of the 29th International Conference on Computational Linguistics. 2022: 7209-7218.

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

[30] Honnibal M, Johnson M. Joint incremental disfluency detection and dependency parsing[J]. Transactions of the Association for Computational Linguistics, 2014, 2: 131-142.

[31] Chen A, Zayats V, Walker D, et al. Teaching BERT to Wait: Balancing Accuracy and Latency for Streaming Disfluency Detection[C]//Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 2022: 827-838.

[32] Weld H, Huang X, Long S, et al. A survey of joint intent detection and slot filling models in natural language understanding[J]. ACM Computing Surveys, 2022, 55(8): 1-38.

[33] Voulodimos A, Doulamis N, Doulamis A, et al. Deep learning for computer vision: A brief review[J]. Computational intelligence and neuroscience, 2018, 2018.

[34] Dong C, Li Y, Gong H, et al. A survey of natural language generation[J]. ACM Computing Surveys, 2022, 55(8): 1-38.

[35] 张晓旭, 马志强, 刘志强, 朱方圆, 王春喻. Transformer 在语音识别任务中的研究现状与展望[J]. 计算机科学与探索, 2021, 15(9): 1578-1594.

[36] 张小峰, 谢钧, 罗健欣, 杨涛. 深度学习语音合成技术综述[J]. 计算机工程与应用, 2021, 57(9): 50-59.

[37] 高璐璐, 赵雯. 机器翻译研究综述[J]. 中国外语, 2020, 17(6): 97-103.

[38] 刘继明, 张培翔, 刘颖, 张伟东, 房杰. 多模态的情感分析技术综述[J]. 计算机科学与探索, 2021, 15(7): 1165-1182.

[39] Diefenbach D, Lopez V, Singh K, et al. Core techniques of question answering systems over knowledge bases: a survey[J]. Knowledge and Information systems, 2018, 55: 529-569.

[40] 冯志伟, 丁晓梅. 计算语言学中的语言模型[J]. 外语电化教学, 2021(06): 17-24+3.

[41] 计峰. 自然语言处理中序列标注模型的研究[D]. 上海: 复旦大学, 2012.

[42] 吴双志, 张冬冬, 周明. 基于自注意力机制的口语文本顺滑算法[J]. 智能计算机与应用, 2019, 9(06): 188-192.

[43] Yoshikawa M, Shindo H, Matsumoto Y. Joint transition-based dependency parsing and disfluency detection for automatic speech recognition texts[C]//Proceedings of the

2016 Conference on Empirical Methods in Natural Language Processing. 2016: 1036-1041.

[44] Rasooli M S, Tetreault J. Joint parsing and disfluency detection in linear time[C]//Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing. 2013: 124-129.

[45] Zayats V, Ostendorf M, Hajishirzi H. Disfluency Detection Using a Bidirectional LSTM[J]. Interspeech 2016, 2016: 2523-2527.

[46] Taylor W L. “Cloze procedure”: A new tool for measuring readability[J]. Journalism quarterly, 1953, 30(4): 415-433.

[47] Lester B, Pressel D, Hemmeter A, et al. Constrained Decoding for Computationally Efficient Named Entity Recognition Taggers[C]//Findings of the Association for Computational Linguistics: EMNLP 2020. 2020: 1841-1848.

[48] Charniak E, Johnson M. Edit detection and parsing for transcribed speech[C]//Second Meeting of the North American Chapter of the Association for Computational Linguistics. 2001.

[49] 苏剑林. 你的 CRF 层的学习率可能不够大[EB/OL]. https://kexue.fm/archives/7196. 2020-2-7.