文本复制检测报告单(全文标明引文)

№:ADBD2019R_2017051613261120190408181456403856835211　　检测时间:2019-04-08 18:14:56

检测文献:　6549072_安磊_基于深度学习的信息抽取服务系统的设计与实现_基于深度学习的信息抽取服务系统的设计与实现

作者:　安磊

检测范围:　中国学术期刊网络出版总库
中国博士学位论文全文数据库/中国优秀硕士学位论文全文数据库
中国重要会议论文全文数据库
中国重要报纸全文数据库
中国专利全文数据库
图书资源
优先出版文献库
学术论文联合比对库
互联网资源(包含贴吧等论坛资源)
英文数据库(涵盖期刊、博硕、会议的英文数据以及德国Springer、英国Taylor&Francis 期刊数据库等)
港澳台学术文献库
互联网文档资源
CNKI大成编客-原创作品库

时间范围:　1900-01-01至2019-04-08

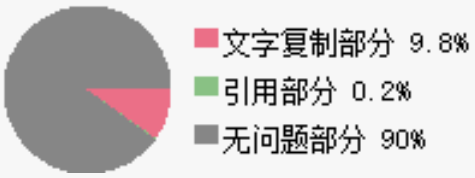
检测结果

去除本人已发表文献复制比：10%　　跨语言检测结果：0%

去除引用文献复制比：9.8%　　总文字复制比：10%

单篇最大文字复制比：4.7%（011_M201570005_聂国平）

重复字数：	[5122]	总段落数：	[12]
总字数：	[51099]	疑似段落数：	[7]
单篇最大重复字数：	[2404]	前部重合字数：	[608]
疑似段落最大重合字数：	[2904]	后部重合字数：	[4514]
疑似段落最小重合字数：	[36]		



指　标：☐疑似剽窃观点　☑疑似剽窃文字表述　☐疑似自我剽窃　☐疑似整体剽窃　☐过度引用

表　格：0　　公　式：等待检测　　疑似文字的图片：0　　脚注与尾注：0

6.8%（195） 6549072_安磊_基于深度学习的信息抽取服务系统的设计与实现_基于深度学习的信息抽取服务系统的设计与实现.doc_第1部分（总2867字）
0%（0） 6549072_安磊_基于深度学习的信息抽取服务系统的设计与实现_基于深度学习的信息抽取服务系统的设计与实现.doc_第2部分（总313字）
0%（0） 6549072_安磊_基于深度学习的信息抽取服务系统的设计与实现_基于深度学习的信息抽取服务系统的设计与实现.doc_第3部分（总441字）
0%（0） 6549072_安磊_基于深度学习的信息抽取服务系统的设计与实现_基于深度学习的信息抽取服务系统的设计与实现.doc_第4部分（总433字）
0%（0） 6549072_安磊_基于深度学习的信息抽取服务系统的设计与实现_基于深度学习的信息抽取服务系统的设计与实现.doc_第5部分（总1561字）
8.6%（494） 6549072_安磊_基于深度学习的信息抽取服务系统的设计与实现_基于深度学习的信息抽取服务系统的设计与实现.doc_第6部分（总5747字）
17%（1256） 6549072_安磊_基于深度学习的信息抽取服务系统的设计与实现_基于深度学习的信息抽取服务系统的设计与实现.doc_第7部分（总7372字）
0.4%（40） 6549072_安磊_基于深度学习的信息抽取服务系统的设计与实现_基于深度学习的信息抽取服务系统的设计与实现.doc_第8部分（总9201字）
7.9%（197） 6549072_安磊_基于深度学习的信息抽取服务系统的设计与实现_基于深度学习的信息抽取服务系统的设计与实现.doc_第9部分（总2480字）
18.4%（2904） 6549072_安磊_基于深度学习的信息抽取服务系统的设计与实现_基于深度学习的信息抽取服务系统的设计与实现.doc_第10部分（总15793字）
0%（0） 6549072_安磊_基于深度学习的信息抽取服务系统的设计与实现_基于深度学习的信息抽取服务系统的设计与实现.doc_第11部分（总3437字）
2.5%（36） 6549072_安磊_基于深度学习的信息抽取服务系统的设计与实现_基于深度学习的信息抽取服务系统的设计与实现.doc_第12部分（总1454字）

（注释：无问题部分 文字复制部分 引用部分）

1. 6549072_安磊_基于深度学习的信息抽取服务系统的设计与实现_基于深度学习的信息抽取服务系统的设计与实现.doc_第1部分

总字数：2867

相似文献列表

去除本人已发表文献复制比：6.8%(195) 文字复制比：6.8%(195) 疑似剽窃观点：(0)

1	基于GeoMedia的高速公路监控系统的研究和应用 刘溯(导师：史金余) -《大连海事大学硕士论文》- 2010-06-01	3.2%（92） 是否引证：否
2	良好亲子关系构建中的社会工作介入研究 刘雨露(导师：张翠娥;钟华) -《华中农业大学硕士论文》- 2012-06-01	2.5%（73） 是否引证：否
3	移动对等网络若干关键技术的研究 牛新征(导师：周明天) -《电子科技大学博士论文》- 2008-09-01	2.4%（68） 是否引证：否
4	智能算法研究及其在网络中的应用 梁淑萍(导师：毛力) -《江南大学硕士论文》- 2012-03-01	2.2%（62） 是否引证：否
5	基于二维码宁波网上购物物流系统的设计与实现 张猛(导师：余堃;高顺林) -《电子科技大学硕士论文》- 2013-09-25	2.2%（62） 是否引证：否
6	微博用户转发预测特征的特征选择研究 赵领帅(导师：管子玉) -《西北大学硕士论文》- 2018-12-01	2.0%（56） 是否引证：否
7	时间序列特征编码方法改进及在金融数据中的应用 董肖凯(导师：李昌峰) -《南京财经大学硕士论文》- 2018-04-01	1.7%（50） 是否引证：否
8	上海某商品房项目施工成本管理研究 李蕾(导师：刘敬孝) -《中国海洋大学硕士论文》- 2013-05-18	1.7%（48） 是否引证：否

原文内容

摘要

自然语言处理作为人工智能领域中的一个重要分支一直是许多人关注的焦点，而知识图谱是自然语言处理中一个重要的研究方向。构建知识图谱的数据来源包括结构化文本和非结构化文本，而如何从非结构化文本中提取高质量的信息是目前研究的热点之一。

针对上文所述的问题，本文设计了一个能够提供高质量数据的信息抽取服务系统。该系统以从非结构化文本中进行信息抽取为背景，以获得构建知识图谱时所需的三元组作为目标，采用了命名实体识别和关系抽取技术，实现了从非结构化文本到知识图谱三元组的数据处理与转换。

本文从模型设计和具体的使用场景出发，融合了两种不同的命名实体识别模型应用和一种关系抽取模型应用，将模型和应用封装在Docker容器中，部署到Kubernetes集群上构建一个信息抽取服务系统。

本文首先介绍了信息抽取的两个子任务命名实体识别和关系抽取的研究背景，阐述了本文的一些主要工作。然后本文对实现信息抽取服务中运用到的一些技术进行简单的介绍与说明。之后根据信息抽取服务系统的架构进行需求分析，包括详细的需求分析、项目的架构分析以及模型设计的详细分析。此后介绍了每个模块的具体实现过程，对所有模型进行实验分析，信息抽取服务进行部署。最后对整个项目进行总结，并对未来的研究工作进行规划与展望。

关键词：命名实体识别，关系抽取，信息抽取服务，非结构化文本，知识图谱

Abstract

Natural Language Processing has always been the focus of many people as an important branch in the field of artificial intelligence, and Knowledge Graph is an important research direction in Natural Language Processing. The data sources for constructing Knowledge Graph include structured texts and unstructured texts, and how to extract high quality information from unstructured texts becomes a research hot spot.

Regarding the issue described above, this thesis designs an information extraction service system that can provide high quality data. Based on the background of information extraction from unstructured texts, this system aims to get triples to construct the Knowledge Graph. The system uses Named Entity Recognition and Relation Extraction techniques to implement data processing and conversion from unstructured texts to Knowledge Graph triples.

Based on the model design and specific usage scenarios, the thesis combine two different Named Entity Recognition model applications and one Relation Extraction model application, package the models and applications in a Docker container and deploy to the Kubernetes cluster to build an information extraction service system.

The thesis firstly introduces the research background of information extraction`s two sub-task Named Entity Recognition and Relation Extraction,explain some main work of this thesis. Then, the thesis briefly introduces and explains some technologies used in the implementation of information extraction service. After that, the thesis describes the requirements analysis, including detailed requirements analysis, project architecture analysis and detailed analysis of model design. Next, the thesis introduces the specific implementation process of each module, conduct experimental analysis on all models and deploy information extraction service. Finally, the thesis summaries the entire project, plans and forecasts the future research work.

Keywords: Named Entity Recognition, Relation Extraction, Deep Learning, Information Extraction Service, Unstructured Text, Knowledge Graph

目录

3. 6549072_安磊_基于深度学习的信息抽取服务系统的设计与实现_基于深度学习的信息抽取服务系统的设计与实现.doc_第3部分

总字数：441

相似文献列表

去除本人已发表文献复制比：0%(0)　　文字复制比：0%(0)　　疑似剽窃观点：(0)

原文内容

4. 6549072_安磊_基于深度学习的信息抽取服务系统的设计与实现_基于深度学习的信息抽取服务系统的设计与实现.doc_第4部分

总字数：433

相似文献列表

去除本人已发表文献复制比：0%(0)　　文字复制比：0%(0)　　疑似剽窃观点：(0)

原文内容

5. 6549072_安磊_基于深度学习的信息抽取服务系统的设计与实现_基于深度学习的信息抽取服务系统的设计与实现.doc_第5部分

总字数：1561

相似文献列表

去除本人已发表文献复制比：0%(0)　　文字复制比：0%(0)　　疑似剽窃观点：(0)

原文内容

表目录

原文内容

第一章绪论

1.1 项目背景

自然语言处理（Nature Language Processing，简称NLP）是指让计算机理解、处理和运用人类语言（如中文、英文等），它属于人工智能（Artificial Intelligence，简称AI）的一个研究方向，是一门计算机科学和语言学的交叉学科。自然语言是人类与其他动物的区别的主要标志，没有语言，人类的思维也就无从可谈，因此NLP代表了人工智能的最高任务与境界。所以只有当计算机具备了处理自然语言的能力时，机器才算实现了真正的智能。近十年，随着计算机硬件的发展、算力极大提升，NLP领域的研究得到了很大的发展，在许多行业都有了广泛的应用。

信息抽取作为NLP中的一项基础任务，在许多上层应用中都有着十分重要的影响。新闻文本中的实体信息是上市公司新闻舆情分析中的关键，如：公司、人、行业等实体信息以及实体间的关系信息，利用信息抽取中的命名实体识别和关系抽取就可以从新闻文本中抽出这些重要信息，进而对新闻舆情进行后续分析；在搜索引擎上进行搜索时的关键信息就是输入的搜索信息中的实体信息，但只通过分词对输入文本进行切分很容易把一个多词组成的命名实体切分成多个词，造成搜索不准确甚至搜索的结果完全不符合预期。如果能对输入文本进行命名实体识别就可以精确识别搜索的内容，提高了搜索的准确率，也提升了检索的效率；在利用非结构化文本信息构建知识图谱的任务中，关键信息就是非结构化文本中的关联实体以及实体间的关系组成的三元组，如：从“阿里巴巴集团的创始人是马云”这段非结构化文本中可以提取出“阿里巴巴，创始人，马云”这个实体-关系-实体的三元组。这里首先用命名实体识别模型识别出了阿里巴巴和马云这两个实体，再用关系抽取对阿里巴巴和马云这两个实体的关系进行识别和抽取。最后，利用从海量非结构化文本中提取出的大量三元组，经过实体消歧、知识融合后构建成一张知识图谱。

在构建上市公司知识图谱时，如何确保实体识别和关系抽取的准确性，提供高效稳定的信息抽取服务是构建高质量、高精度的知识图谱的关键所在。本文同时结合了命名实体识别和关系抽取两项任务，并将这两种模型进行组合，搭建了一个信息抽取服务系统来对非结构化文本中的三元组信息进行抽取。

命名实体识别（Named Entity Recognition，简称NER）作为信息抽取中的一个基本任务，最早是在MUC-6（the Sixth of the Message Understanding Conference）上将其作为信息抽取的一项子任务引入测评中[MUC-6, 1996]。关系抽取（Relation Extraction，简称RE）作为信息抽取中的另一个基本任务，早在2000年就有基于句法解析增强的方法来实现关系抽取任务[Miller et al., 2000]。

1.2 国内外研究现状

1.2.1 命名实体识别研究现状

自从在MUC-6 [MUC-6, 1996]上将命名实体识别作为信息抽取的一个子任务之后，命名实体识别就成为了从事信息抽取领域的专家学者们研究的焦点之一。而随着数据时代的到来，命名实体识别在各个领域的信息抽取任务中更是得到了充分的发展。

在国外命名实体识别的研究中，考虑到英文本身的特性，英文的命名实体识别只需要考虑词本身的特征而不需要关注如分词等问题的影响，所以实现的难度相对较小，准确率也较高，

中文命名实体识别由于中文文字内在的特殊性导致了在做命名实体识别之前要先进行分词、句法分析等操作，使中文命名实体识别相对英文而言难度大大提高。在MUC-6上国内专家学者们把“人名”、“地名”、“组织机构名”作为实体的类别，将这三类实体作为主要研究目标。随着研究的不断深入，逐渐有人将“时间”、“货币”、“百分数”等也作为实体类别加入到命名实体识别任务中。

除此之外，中文命名实体识别相对于英文命名实体识别任务还存在许多其他的不同之处，由于中文本身的特点造成了许多的难点：

（1）中文单词的边界相比于英文较模糊

英文中的单词之间都有分隔符来标识边界，命名实体如人名、地名等单词的首字母为大写，这些信息能很好地为命名实体的识别做边界和位置标识，而中文单词没有这些信息。

（2）中文命名实体的词语结构更加复杂

有的类型的命名实体单词长度没有限制；不同实体有不同的组成结构（嵌套、简称、别名等）；音译词没有统一的构词规范；组织机构名和人名、地名会有很多交叉重复的地方（如：xxx（上海）有限公司）。

（3）中文命名实体的一词多义与一义多词

随着互联网的极速发展，越来越多的网络词汇不断出现，很多命名实体都有了新的称呼，如称淘宝为某宝。这些新称呼的出现大大增加了命名实体识别的难度。

在早期的命名实体识别任务中，主要有构建基于专家知识的规则以及基于数理统计两类方法。基于规则的方法主要是依靠人工构建规则体系，在构建了大量基于专家知识的词法和语义规则之后，系统会对输入文本进行基于规则的解析，对文本中可能的命名实体进行推理和识别。虽然在特定的数据集上基于专家知识构建规则比基于统计方法的命名实体识别准确度更高，但这种方法在数据量逐渐增大、数据内容逐渐复杂之后会变得不再可行，因为基于某一小部分语料构建的规则体系在别的语料上并不适用，人们无法扩充和维护一个十分庞大的规则体系。随着机器学习理论的完善和计算机性能的提高，大部分学者开始转而研究基于统计方法的命名实体识别。

基于统计方法的命名实体识别主要为有监督（Supervised）和半监督（Semi-Supervised）的机器学习方法。有监督的机器学习方法在拥有大量标注语料的前提下，在各个领域的文本中都有着更高的识别准确率，从而受到更多的学者、从业者们的青睐。在命名实体识别任务中表现比较出色的有监督学习方法主要有：隐马尔科夫模型（Hidden Markov Models，简称HMM）[Bikel et al., 1999]、最大熵模型（Maximum Entropy Models，简称MEM）[Tsai et al., 2004]、条件随机场（Conditional Random Fields，简称CRF）[McCallum et al., 2003]。基于统计方法的命名实体识别对特征选取的要求比较高

，需要从文本中挖掘出对于实体识别有用的单词信息、上下文信息、句法信息、语义信息等作为特征。随着研究的不断深入，大量实验结果表明CRF结合了HMM和MEM的优点，成为中文命名实体识别任务中表现更优秀的统计学习方法。有学者针对CRF的特征选择与交叉组合进行了研究，通过实验得出不同特征以及组合特征在训练时对模型预测效果的贡献大小[张祝玉等, 2008]。

随着深度学习的影响不断增加，用深度学习方法解决命名实体识别任务也取得了一些显著的成果。[Wu et al., 2015]提出使用深度神经网络（Deep Neural Network，简称DNN）从语料中训练词向量，再将词向量输入到另一个深度神经网络中进行命名实体识别，实验结果好于传统机器学习中效果最好的CRF模型。[Z Huang et al., 2015]提出结合双向长短时记忆模型(Bidirectional Long-Short Term Memory Model，简称Bi-LSTM)和条件随机场进行命名实体识别。还有很多深度学习方法对命名实体识别任务做了尝试，在不同的数据集上相较于传统方法都有不小的提升。在NLP任务中，通常会先使用预训练好的词向量对输入进行词嵌入操作，因为预训练好的词向量可以提取到词级别的特征，有助于提升后续任务的效果。Google在2018年提出的BERT模型[Devlin J et al., 2018]作为NLP任务中的预训练模型，结合对下游任务的微调在NLP的各项任务中的表现都达到了SOTA（state-of-the-art，最先进水平的）。表1.1为目前命名实体识别的一些常用方法汇总。

表 1.1 常用命名实体识别方法汇总

模型名称备注

CRF 效果最好的传统机器学习方法

word2vec+DNN 深度神经网络作为网络层

word2vec+LSTM 单向LSTM作为网络层

word2vec+Bi-LSTM 双向LSTM作为网络层

word2vec+LSTM+CRF 单向LSTM结合CRF进行分类

word2vec+Bi-LSTM+CRF 双向LSTM结合CRF进行分类

BERT+Bi-LSTM+CRF 利用BERT作为词嵌入模型

1.2.2 关系抽取研究现状

从对标注数据的依赖程度可以将关系抽取的方法分为以下三类：

（1）有监督的学习方法

把关系抽取任务当做一个分类问题，从训练数据中提取有效的特征并训练各种分类模型进行关系预测。有监督学习可以从标注好的训练数据中学习到各种特征，但是这种方法需要大量的人工标注语料，比较耗时耗力，但预测效果较好。

（2）半监督的学习方法

主要是用Bootstrapping方法进行关系抽取。先设定若干种子实例，再迭代地从文本中抽取相应的关系模板和更多关系实例。

（3）无监督的学习方法

无监督学习是假设有相同语义的命名实体且拥有相似的上下文信息，可根据每个命名实体对的上下文信息来代表它们的语义关系，并对所有实体对的语义进行聚类。

自从2000年Miller等提出基于句法解析增强的方法来实现关系抽取[Miller et al., 2000]后，越来越多的实体间关系抽取方法被提出：基于逻辑回归的方法[Kambhatla et al., 2004]、基于核函数的方法[Zhao and Grishman, 2005]、基于条件随机场的方法[Culotta et al., 2006]。在有监督学习中针对需要大量人工标注的情况，Mintz等人[Mintz et al., 2009]提出了使用远程监督（Distant Supervision）的方法来扩充标注语料，可以有效解决关系抽取的标注数据规模问题。

随着计算机计算能力的发展，很多学者也开始将深度学习方法运用到关系抽取任务中。[Socher et al., 2012]提出使用递归神经网络（Recurrent Neural Network，简称RNN）来解决关系抽取问题，通过递归神经网络学习到句子的词汇特征、句法特征、语义特征再用于关系分类和抽取。[Zeng et al., 2014]提出使用卷积神经网络（Convolutional Neural Network，简称CNN）来解决关系抽取问题，采用词向量和词的相对位置作为卷积神经网络的输入，通过卷积、池化、非线性计算等操作得到句子特征并用于关系抽取。[Santos et al., 2015]提出了一种新的卷积神经网络结构用于解决关系抽取问题，在这个新的结构中采用了新的Ranking损失函数并得到了更好的效果。[Miwa er al., 2016]提出一种基于端到端（End to End）神经网络的关系抽取模型，使用双向长短时记忆模型和树形长短时记忆模型同时对实体和句子进行特征提取与建模。[Lin et al., 2016]提出基于句子级注意力机制的神经网络模型来解决关系抽取问题，文中的方法可以根据不同关系为每个实体对分配不同的权重。表1.2所示为目前中文关系抽取常用的一些方法汇总。

表1.2 常用关系抽取方法汇总

模型名称备注

RNN 循环神经网络作为网络层

CNN 卷积神经网络作为网络层

CNN+Attention 卷积神经网络结合注意力机制

Bi-LSTM+Attention 双向LSTM结合注意力机制

PCNN+Attention 带位置信息的卷积神经网络结合注意力机制

ResCNN 卷积神经网络结合残差学习

1.3 本文主要的工作

本文在总结信息抽取现有的一些研究成果的基础上，结合命名实体识别和关系抽取两项信息抽取的基本任务，从多个深度学习模型中选择合适的模型，利用人工标注的部分金融领域新闻文本语料结合远程监督进行模型训练，将训练好的模型固化成Protocol Buffers（简称pb）文件，把模型进行组合封装到Docker容器中，并部署在集群上来提供信息抽取服务。

本文的主要工作有：

1. 对命名实体识别和关系抽取的研究现状和相关技术进行综述，对命名实体识别模型和关系抽取模型进行实验对比分析。

2. 对本文的信息抽取服务系统进行了详细设计，包括需求分析、产品功能分析、本文选择的信息抽取模型的详细设计以

及系统中每个模块的详细设计。

3. 将训练好的模型包成app，封装到Docker容器中并部署到Kubernetes集群上，构建一套信息抽取服务系统。

1.4 本文的组织结构

本文的各章节内容安排如下：

第一章绪论。本章介绍了论文的项目背景，信息抽取任务中的两个子任务命名实体识别和关系抽取，国内外在这两个方向上的研究成果，本文所做的主要工作和论文的组织结构。

第二章技术综述。本章详细介绍了论文涉及的相关技术，包括词向量、BERT模型、双向长短时记忆模型、深度残差网络、远程监督、深度学习框架Tensorflow、Docker容器技术、Kubernetes集群、Scrapy爬虫等。

第三章信息抽取服务分析与设计。本章详细介绍了信息抽取服务的总体规划、信息抽取服务的需求分析、信息抽取服务中各个模块的详细设计以及信息抽取模型结构的设计。

第四章信息抽取服务的实现。本章阐述了信息抽取服务中各个模块的详细实现，包括数据预处理模块、信息抽取模型模块、信息抽取服务模块。

第五章实验对比与分析。本章主要通过实验介绍了项目中选择的模型在不同参数下的模型效果对比以及不同模型之间的性能对比。

第六章总结与展望。总结在论文期间做了哪些工作，在信息抽取任务中未来的可改进的地方以及对接下来的需要做的工作的进一步展望。

指　标
疑似剽窃文字表述

1. 1.4 本文的组织结构
   本文的各章节内容安排如下：
   第一章绪论。本章介绍了论文的

7. 6549072_安磊_基于深度学习的信息抽取服务系统的设计与实现_基于深度学习的信息抽取服务系统的设计与实现.doc_第7部分

总字数：7372

相似文献列表

去除本人已发表文献复制比：17%(1256)　　文字复制比：17%(1256)　　疑似剽窃观点：(0)

1	11881446_肖朴_基于大数据的网络安全态势预测方法研究 肖朴 - 《学术论文联合比对库》- 2017-10-28	3.8%（282） 是否引证：否
2	Docker与KVM之间的区别 - huaweitman的专栏 - 博客频道 - CSDN.NET -《网络（http://blog.csdn.net）》- 2017	3.7%（271） 是否引证：否
3	Docker整理之简介（一）- timeless的博客 - 博客频道 - CSDN.NET -《网络（http://blog.csdn.net）》- 2017	3.7%（271） 是否引证：否
4	云计算与大数据背后的核心技术研究 -《学术论文联合比对库》- 2017-09-04	3.6%（265） 是否引证：否
5	【深入浅出容器云】关于容器云你不得不知的十大特性 - 云计算 -《网络（http://www.ciotimes.）》- 2016	3.5%（258） 是否引证：否
6	2_肖朴_基于大数据的网络安全态势预测方法研究 肖朴 - 《学术论文联合比对库》- 2017-10-16	3.4%（247） 是否引证：否
7	docker【1】docker简介（入门知识）- 既认准这条路，又何必在意要走多久 - 博客频道 - CSDN.NET -《网络（http://blog.csdn.net）》- 2017	3.3%（245） 是否引证：否
8	云计算 -《学术论文联合比对库》- 2015-09-30	3.3%（242） 是否引证：否
9	信息学院-张晓燕 张晓燕 - 《学术论文联合比对库》- 2017-10-10	2.8%（205） 是否引证：否
10	11942841_肖朴_基于大数据的网络安全态势预测方法研究 肖朴 - 《学术论文联合比对库》- 2017-12-01	2.7%（198） 是否引证：否
11	Docker容器及Spring Boot微服务应用 - 小飞侠的博客 - 博客频道 - CSDN.NET -《网络（http://blog.csdn.net）》- 2017	2.7%（196） 是否引证：否
12	k8s docker集群搭建 -《互联网文档资源（https://wenku.baidu.）》- 2018	2.2%（160） 是否引证：否
13	“智慧校园”-教育服务云平台的研究 王芳(导师：赵振东;程建军) -《华北电力大学硕士论文》- 2018-06-01	1.8%（130） 是否引证：否

14	1334061002_王耄_081202_计算机软件与理论_王晓明 王耄 - 《学术论文联合比对库》- 2016-04-23	1.5%（113） 是否引证：否
15	1334061002_王耄_081202_计算机软件与理论_王晓明 王耄 - 《学术论文联合比对库》- 2016-04-23	1.5%（113） 是否引证：否
16	Kubernetes将统治云端 张迟第 - 《人民邮电》- 2017-12-14	1.3%（97） 是否引证：否
17	基于Word2Vec的主题爬虫研究与实现 宋健(导师：赫枫龄) - 《吉林大学硕士论文》- 2018-04-01	1.2%（89） 是否引证：否
18	社区问答系统中的专家推荐方法研究 孙吉庆(导师：王健) - 《大连理工大学硕士论文》- 2017-05-01	1.0%（74） 是否引证：否
19	基于深度学习的中文词表示学习技术研究 庄航(导师：周学海) - 《中国科学技术大学博士论文》- 2018-08-22	0.9%（64） 是否引证：否
20	浅析基于微服务架构的测试云平台的移动应用兼容性测试实现 张倩;刘侃;周宇; - 《科技资讯》- 2018-10-03	0.7%（49） 是否引证：否
21	基于深度学习的森林资源信息估测模型研究 汪康宁(导师：马庆勋) - 《西安科技大学硕士论文》- 2018-06-01	0.7%（49） 是否引证：否
22	基于深度学习的三维模型补全技术研究 满婧琦(导师：魏小鹏) - 《大连理工大学硕士论文》- 2018-06-08	0.7%（48） 是否引证：否
23	基于事件框架的生物信息抽取的研究 王安然(导师：王健) - 《大连理工大学硕士论文》- 2018-05-01	0.4%（29） 是否引证：否

原文内容

第二章技术综述

本章主要介绍了本文中实现的基于深度学习的命名实体识别模型和关系抽取模型所涉及的相关技术，包括词向量、BERT模型、双向长短时记忆模型、深度残差网络、远程监督、深度学习框架Tensorflow、Docker容器技术、Kubernetes集群、Scrapy爬虫等。下面将对这些技术的特点和实际应用场景分别进行介绍。

2.1 词向量

因为计算机不能直接识别自然语言，所以要想让计算机能够理解自然语言，必须要对自然语言进行一些处理，因此词向量（Word Embedding）的概念被提出。计算机的通用语言是二进制数，词向量就是一些包含了自然语言中词汇和语义信息的数字。

词表示方法最早是使用One-Hot编码，即用N个状态编码器对N个状态进行编码，这种方法会造成一篇篇幅有n个字的文本，会用一个n*n的矩阵来表示，每个n维向量中只有对应的那个字的位置上是1，其余位置都是0。这种词表示方法在传统机器学习中，如条件随机场、逻辑回归、最大熵、支持向量机等模型中可以较好的完成一些自然语言处理的任务。但是这种词表示方法缺点在于默认每个单词是独立存在的，从而忽略了文本中单词间的语义关联，且通常向量的维度会很高，在用深度学习处理自然语言处理问题时如果使用One-Hot编码容易造成“维度灾难”[Bengio et al., 2002]。

除了One-Hot编码，基于[Harris, 1954]提出的分布假说：“上下文相似的词，其语义也相似。”[Hinton, 1986]提出了一种新的词表示方法：分布式表示法（Distributed Reputation），即用向量来表示一个词。词的分布式表示主要分为两类：基于矩阵的分布表示和基于神经网络的分布表示。

2.1.1 基于矩阵的分布表示

基于矩阵的分布表示主要是构建“词-上下文”矩阵，从矩阵中获取词的分布表示。矩阵的每一行都表示一个词，列表示上下文信息。常见的上下文有：

（1）文档，即“词-文档”矩阵。

（2）上下文的单词，即“词-词”矩阵。

（3）n-元词组，即“词-n-元词组”矩阵。

矩阵中的每个元素表示词和上下文共现的次数，可以利用词频-逆文本频率指数（TF-IDF）、取对数等技巧进行加权和平滑。矩阵维度较高且较稀疏的情况还可以利用奇异值分解（SVD）和非负矩阵分解（NMF）等方法进行分解降维使之变成低维稠密矩阵。基于矩阵的分布表示的典型代表是Global Vector模型（GloVe模型）[Pennington et al., 2014]。

2.1.2 基于神经网络的分布表示

基于神经网络的分布表示一般通过神经网络训练语言模型得到，[Bengio et al., 2006]提出了一种神经网络语言模型（Neural Network Language Model，简称NNLM），考虑对语言模型进行建模。但是NNLM中词向量只是语言模型训练得到的副产物，并没有指出哪一套向量作为词向量的效果更好，所以[Mikolov et al., 2013a; Mikolov et al., 2013b]在NNLM等模型的基础之上，提出了word2vec模型，包含CBOW（Continuous Bag-of-Words）和Skip-gram两个模型，实现了高效地获取词向量，CBOW和Skip-gram的示意图如图2.1所示。

图2.1 CBOW和Skip-gram示意图

CBOW模型是根据上下文的信息来预测中间的目标词。CBOW模型对NNLM模型做了简化：隐藏层不再是上文各词词向量的拼接，而是使用上下文各词词向量的平均值，减少了计算量；去掉了tanh隐藏层，提升了模型的训练速度。CBOW模型包含3层：输入层、投影层和输出层。输入层共有n-1个词的One-Hot表示作为输入，组成上下文的表示：。投影层将输入层的n-1个向量做均值计算：。输出层一共V个节点，第i个节点表示中心词是词的概率。CBOW模型根据上下文的表示，直接对目标词进行预测：。对于整个语料库而言，CBOW的优化目标为最大化：

Skip-gram模型与CBOW模型刚好相反，它是根据一个单词来预测它的上下文信息。Skip-gram模型训练一个双层神经网

络，模型输入为一个词，来预测这个词的前n个词和后n个词出现的概率。Skip-gram和CBOW一样没有隐藏层。Skip-gram的优化目标为最大化：

2.1.3 词向量的使用

因为本文所使用的编程语言是Python，Python编程环境中有Gensim库可以直接调用models.word2vec来训练词向量。[Radford et al., 2017]提出使用任务相关的语料训练得到的词向量相对于别的词向量在特定任务中会有更好的效果。由于本文所涉及的项目是针对金融领域的文本进行信息抽取且命名实体识别的序列标注任务是针对每个字来进行标注的，所以使用金融相关文本作为训练字向量的文本语料，训练了一个包含6000个字的100维的字向量模型。

2.2 BERT模型

一般的词向量都是通过浅层网络进行无监督训练得到的，虽然在词的级别上有着不错的特性，但是缺少对连续文本的内在联系和语言结构的表达能力。因此希望通过大量数据来预训练一个大型的神经网络，用它来对文本进行特征提取之后再去做后续的任务，以期望得到更好的效果。

2018年Google提出的BERT模型结合了当时所有的预训练模型的优点，并采用了大量的数据和更大的模型来进行预训练，得到了当时在业界中表现最好的预训练模型。

BERT模型主要使用Transformer编码器作为主体模型结构。Transformer是一种完全舍弃了RNN的循环式网络结构的，完全基于注意力机制的网络结构模型。图2.2所示为Transformer编码器的结构图。

图2.2 Transformer编码器结构图

Transformer编码器的核心思想是计算一句话中每个词对于这句话中所有词的相互关系，认为这些词与词之间的相互关系在一定程度上反映了这句话中不同词之间的关联性和重要程度，再利用这些相互关系来调整每个词的权重来获得每个词的新的表征。这个新的表征不但蕴含了词本身的含义，还蕴含了这个词与其他词的关系，相比于词向量，这样的词表征所包含的信息更加全面。Transformer就是通过对输入的文本不断进行这样的注意力机制层和普通非线性层交叠来得到最终的文本表征。

BERT模型主要的创新在于使用MaskLM的方式来训练语言模型以及增加句子级别的连续性预测任务“next sentence prediction”。MaskLM是指随机遮挡输入中的一些tokens并用一些特殊的符号进行替代，然后在预训练中对他们进行预测。BERT模型为了和后续任务保持一致，按一定比例在需要预测的词的位置上输入原词或者输入某个随机的词。由于一次输入的文本只有部分词被用来训练，所以BERT在效率上会较低，收敛会需要更多的训练步数；“next sentence prediction”是指预测输入文本是否为连续的文本。在训练时输入模型的第二个片段有50%的概率从全部文本中随机选择一句话，还有50%的概率选择第一段文本的后续文本。引入这个任务是为了让模型更好地学习连续的文本片段之间的关系。

2.3 双向长短时记忆模型

2.3.1 循环神经网络

循环神经网络（Recurrent Neural Networks，简称RNN）[Rumelhart et al. 1986]指的是一个序列当前的输出和之前的输出存在某种关联的神经网络。具体的表现形式为网络会对前面的信息进行记忆，保存在网络的内部状态中，并应用于当前输出的计算中。所以循环神经网络常常被用于对序列数据进行建模。循环神经网络的结构图如图2.3所示。

图2.3 循环神经网络的结构图

理论上循环神经网络能够对任何长度的序列数据进行处理，但是在实践中发现”梯度消失”会影响循环神经网络的训练，随着神经网络层数的增加，“梯度消失”的风险就会越大，当层数很大时，底层的神经元将接收不到返回的信号，出现“梯度消失”的问题。而长短时记忆模型（Long-Short Term Model，简称LSTM）[Hochreiter et al., 1997]的出现缓解了这个问题。

2.3.2 长短时记忆模型

长短时记忆模型的提出是为了解决循环神经网络的“梯度消失”问题，它之所以能够缓解这个问题主要取决于LSTM特有的门型结构，如图2.4所示。

图2.4 LSTM结构示意图

可以看到，LSTM主要包括遗忘门、输入门和输出门等结构，其中：

遗忘门：

输入门：

输出门：

新记忆单元：

最终记忆单元：

隐藏层：

三个不同的门型结构和记忆单元是LSTM的核心所在，记忆单元负责保存历史和当前的重要信息，遗忘门负责控制选择性丢弃之前的记忆单元中的一些信息，输入门负责控制当前的输入中的某些信息加入到记忆单元中，输出门负责控制记忆单元中的哪些信息可以进行输出。这些门控结构可以帮助LSTM在长时和短时的记忆中都能处理并解决数据依赖以及选择是否丢弃信息（保留重要信息并丢弃不重要的信息）。三个门相互合作是LSTM能够解决RNN梯度消失/梯度爆炸的关键所在。

2.3.3 双向长短时记忆模型

一般我们所说的RNN、LSTM都是单向的，也就是说在一个方向上进行隐藏状态的传递，然而在某些任务中需要考虑的不仅仅是当前输入单词前面的文本信息，还需要考虑当前输入单词后面的文本信息。双向长短时记忆模型就可以解决这种类型的问题，它不仅可以从前向后传递隐藏状态的信息，还可以从向前传递隐藏状态的信息，这样在计算输出时考虑的就是完整的上下文信息。可以把双向长短时记忆模型看成是两个方向相反的LSTM的叠加，在分别做计算后把两个状态向量拼接得到最后的输出。

2.4 深度残差网络

深度学习中对于网络深度加深遇到的主要问题是梯度消失和梯度爆炸，传统对应的解决方案是数据的初始化（normalized initialization）和（batch normalization）正则化。

深度残差网络（Deep Residual Network，简称DRN）[He K M. et al. 2015]的提出是源于一个“反常”的现象：在训练深度神经网络的时候，训练误差和测试误差都随着网络深度的增加而增加（Degradation）。在自然语言处理中，利用网络结构训

练文本数据，浅层的网络更容易学习到文本中的词汇信息，而深层的网络更容易学到文本中的语义信息，但是由于层数的增加会导致训练误差的累积，残差网络将浅层的网络与深层的网络进行级联再输出，很好地控制了训练误差的累积，在自然语言处理的各项任务中取得了很好的效果。

假设深度网络中的某隐藏层为，如果多个非线性层组合可以近似于一个复杂函数，那么同样可以认为隐藏层的残差近似于某个复杂函数，那么可以将隐藏层表示为。这样一来就可以得到一种全新的残差结构单元，如图2.5所示。

图2.5 残差结构单元示意图

残差单元的输出由多个卷积层级联的输出和输入元素相加（维度相同的情况下），再经过ReLU激活后得到。将多个这样的结构级联起来，就得到了残差网络。

可以注意到残差网络有这样几个特点：1.网络较瘦，控制了参数数量；2.存在明显层级，特征图个数逐层递进，保证输出特征表达能力；3.使用了较少的池化层，大量使用下采样，提高传播效率；4.没有使用Dropout，利用BN和全局平均池化进行正则化，加快了训练速度。

2.5 远程监督

远程监督（Distant Supervision）是目前关系抽取中比较常见的一种做法。它既不是传统意义上的监督学习，也不是无监督学习，它属于一种用KB（Knowledge Base）去对齐朴素文本的标注方法：1.使用NET（Named Entity Tagger）标注；2.对已标注的金融领域文本中出现的三元组提取特征（从所有出现该三元组的句子中），构造训练数据；3.采用多类别逻辑回归进行分类。

在测试使用时，先使用命名实体识别模型对目标文本中的命名实体进行标注，抽取其中的命名实体对和特征。如果多个句子的命名实体对一样，则把它们的特征合并在同一个特征向量中，再利用逻辑回归分类器对关系名称进行识别。

简单来说，如果训练语料中的句子所包含的实体对在已有的KB中有关系的体现，则认为语料库中所包含相同实体对的句子都表达相应关系。虽然这样做可以减少关系抽取任务对人工标注数据的依赖，但是会引入很多噪声数据。因为两个实体之间可能有别的关系（不同于KB中已有的关系）或者是没有关系，这样的训练数据就会对我们的关系抽取任务产生一定影响。

近些年来，也有很多国内外学者在研究如何降低远程监督中的噪声数据对关系抽取造成的影响，如使用Multi-instance从训练集中抽取置信度较高的训练样例来训练模型、利用Attention机制对数据进行全方位的权重计算从而得到全面而不失“选择”的训练数据等方法。

2.6 其他相关技术

本节主要介绍了除了模型外的一些项目中用到的技术，比如开源深度学习框架Tensorflow，Flask框架，Docker容器技术以及Kubernetes集群等。

2.6.1 深度学习框架Tensorflow

Tensorflow既是一个实现机器学习算法的接口，也是执行机器学习算法的框架。它前端支持Python、C++、Go、Java等多种开发语言，后端使用C++、CUDA等写成。除了执行深度学习算法，Tensorflow还可以用来实现很多传统机器学习算法，如逻辑回归、随机森林等。

Tensorflow使用数据流式图来规划计算流程，可以把计算映射到不同的硬件甚至是操作系统。Tensorflow的计算可以表示为有状态的数据流式图，可以让用户简单地实现并行计算，同时使用不用的硬件资源进行训练，同步或异步地更新全局共享的模型参数和状态。

2.6.2 Docker容器技术

Docker是一个开源的引擎，可以轻松地为任何应用创建一个轻量级的、可移植的、自给自足的容器。开发者编译测试通过的容器可以批量地在生产环境中部署，包括VMs（虚拟机）、bare metal、OpenStack集群和其他基础应用平台。

Docker跟传统的虚拟化方式相比具有很多优势，其优势主要体现在：

1. 更高效的虚拟化：Docker容器的运行不需要额外的hypervisor支持，它是内核级的虚拟化，因此可以实现更高的性能和效率。
2. 更安全的运行应用：Docker容器内运行的进程完全与系统隔离，一些恶意行为对系统造成的影响不会波及宿主系统。
3. 更快速的交付和部署：开发者可以使用一个标准的镜像来构建一套开发容器，开发完成之后，运维人员可以直接使用这个容器来部署代码、Docker可以快速创建容器，快速迭代应用程序，并使整个过程全程可见，使团队其他成员更容易理解应用程序是如何创建和工作的。
4. 更轻松的迁移和扩展：Docker容器几乎可以在任意平台上运行，包括物理机、虚拟机、公有云、私有云、服务器等。这种兼容性可以帮助用户很轻松地将应用程序从一个平台直接迁移到另一个平台上。

2.6.3 Kubernetes集群

Kubernetes（简称k8s）是一个自动化容器操作的开源平台，这些操作包括部署、调度和节点集群间扩展。上一节中介绍的Docker容器就可以看作是Kubernetes内部使用的低级别组件。Kubernetes的主要组件有：

1. etcd:高可用存储共享配置和服务发现，作为与minion机器上的flannel配套使用，使每台minion上运行的docker拥有不同的ip。
2. flannel：网络结构支持。
3. kube-apiserver：无论通过kubectl还是使用remote api直接控制，都要经过apiserver。
4. kube-controller-manager：与apiserver交互，保证controller的工作。
5. kube-scheduler：根据特定的调度算法将pod调度到指定的工作节点上。
6. kubelet：container agent的逻辑继任者，运行在工作节点上。
7. kube-proxy：运行在minion节点上的一个组件，是一个服务代理。

2.6.4 Scrapy爬虫

本文的数据爬虫利用Scrapy爬虫框架实现。Scrapy爬虫框架的架构图如图2.6所示。

图2.6 Scrapy爬虫框架架构图

其中Scrapy Engine（引擎）负责所有模块的中间通讯、信号、数据传递；Scheduler（调度器）负责接收引擎发来的

Request请求，按照一定的方式整理排列、入队，当引擎需要的时候交还给引擎；Downloader（下载器）负责下载引擎发送的所有Request请求，将获取到的Response交还给引擎，再由引擎交给Spider进行后续处理；Spider（爬虫）负责处理所有Responses，分析并从中提取数据，将需要跟进的URL提交给引擎再次进入Scheduler；Item Pipline（管道）负责处理Spider中获取的Item，并进行后续处理，如数据分析、过滤、存储等；Middlewares（中间件）是一个可以自定义扩展功能的组件。

2.7 本章小结

本章主要介绍了本文对信息抽取中的命名实体识别和关系抽取的研究中涉及的相关技术。先是介绍了作为模型输入的深度学习自然语言问题的基础：词向量。之后又分别介绍了BERT模型、双向长短时记忆模型、深度残差网络、远程监督等，并介绍了本文实现模型使用的深度学习框架Tensorflow、Docker容器技术、Kubernetes集群、Scrapy爬虫等。

指　标

疑似剽窃文字表述

---

1. 基于矩阵的分布表示
   基于矩阵的分布表示主要是构建“词-上下文”矩阵，从矩阵中获取词的分布表示。矩阵的每一行都表示一个词，
2. 具体的表现形式为网络会对前面的信息进行记忆，保存在网络的内部状态中，并应用于当前输出的计算中。
3. Tensorflow
   Tensorflow既是一个实现机器学习算法的接口，也是执行机器学习算法的框架。它前端支持Python、C++、Go、Java等多种开发语言，后端使用C++、CUDA等写成。除了执行深度学习算法，Tensorflow还可以用来实现很多传统机器学习算法，如逻辑回归、随机森林等。
   Tensorflow使用数据流式图来规划计算流程，可以把计算映射到不同的硬件甚至是操作系统。
4. 可以让用户简单地实现并行计算，同时使用不用的硬件资源进行训练，同步或异步地更新全局共享的模型参数和状态。
   2.6.2 Docker容器技术
   Docker是一个开源的引擎，可以轻松地为任何应用创建一个轻量级的、可移植的、自给自足的容器。
5. 高效的虚拟化：Docker容器的运行不需要额外的hypervisor支持，它是内核级的虚拟化，因此可以实现更高的性能和效率。
   2. 更安全的
6. 开发者可以使用一个标准的镜像来构建一套开发容器，开发完成之后，运维人员可以直接使用这个容器来部署代码、Docker可以快速创建容器，快速迭代应用程序，并使整个过程全程可见，使团队其他成员更容易理解应用程序是如何创建和工作的。
7. 轻松的迁移和扩展：Docker容器几乎可以在任意平台上运行，包括物理机、虚拟机、公有云、私有云、服务器等。这种兼容性可以帮助用户很轻松地将应用程序从一个平台直接迁移到另一个平台上。
8. Docker容器就可以看作是Kubernetes内部使用的低级别组件。Kubernetes
9. 组件有：
   1. etcd:高可用存储共享配置和服务发现，作为与minion机器上的flannel配套使用，使每台minion上运行的docker拥有不同的ip。
10. -apiserver：无论通过kubectl还是使用remote api直接控制，都要经过apiserver。
    4. kube-controller-mana

8. 6549072_安磊_基于深度学习的信息抽取服务系统的设计与实现_基于深度学习的信息抽取服务系统的设计与实现.doc_第8部分

总字数：9201

相似文献列表

去除本人已发表文献复制比：0.4%(40)　　文字复制比：0.4%(40)　　疑似剽窃观点：(0)

1	基于迁移学习的无参考视频质量评价 张浩;桑庆兵; -《激光与光电子学进展》- 2018-04-27 1	0.4%（40） 是否引证：否

原文内容

第三章信息抽取服务分析与设计

本章通过分析信息抽取服务的具体使用场景，结合业务人员的意见对信息抽取服务进行了模块划分以及详细的需求描述，又对信息抽取服务系统进行了总体规划，对服务中的每个模块进行了详细设计。

3.1 信息抽取服务需求分析

3.1.1 功能性需求

信息抽取服务的系统用例图如图3.1所示。开发人员负责数据预处理、模型训练、模型测试、信息抽取服务维护等，而业务人员主要是信息抽取服务的用户，这一类人员主要需要选择信息抽取的模型组合以及调用服务。

图3.1 系统用例图

本信息抽取服务系统的开发人员需要用到的功能有：数据预处理、模型训练

、模型测试、信息抽取服务。系统的业务人员需要用到的功能有：模型选择和信息抽取服务。

表3.1所示为数据预处理部分的用例描述。数据预处理模块在接收到文本后进行数据处理，包括不可用信息的过滤以及词

向量的训练。

表3.1 数据预处理用例描述

用例编号 UC01

用例名称数据预处理

用例描述对输入文本进行数据清洗，过滤不可用信息；训练词向量模型

参与者开发人员

触发条件开发人员执行数据预处理

前置条件非结构化文本数据传入数据预处理模块

后置条件数据预处理完成；词向量模型训练完毕

优先级高

正常流程开发人员传入非结构化文本数据，执行数据预处理命令系统对输入文本数据进行数据处理并保存系统利用处理好的数据进行词向量训练并保存

扩展流程 3a.检测到已经存在词向量 (1)提示开发人员已经存在词向量，不再训练，直接读取词向量

表3.2 模型训练用例描述

用例编号 UC02

用例名称模型训练

用例描述开发人员设置模型参数，进行模型训练，训练结束后保存模型

参与者开发人员

触发条件开发人员设置模型参数，执行模型训练

前置条件模型训练代码编写完成

后置条件模型训练成功并保存

优先级高

正常流程开发人员设置模型参数，执行模型训练开发人员根据模型训练时打印的日志观察P、R、F1值和损失值，了解模型训练情况模型训练完成并保存模型

表3.2所示为模型训练部分的用例描述，在数据预处理过后，开发人员可以设置不同的参数进行模型训练，训练结束后保存训练的模型。

表3.3所示为模型测试部分的用例描述，在模型训练后，开发人员可以使用保存下来的模型在测试集上进行测试，根据模型在测试集上的预测性能和预测速度来测试模型的整体性能。

表3.3 模型测试用例描述

用例编号 UC03

用例名称模型测试

用例描述开发人员用训练好的模型在测试集上进行测试

参与者开发人员

触发条件开发人员执行模型测试

前置条件模型训练完成

后置条件模型测试完成

优先级高

正常流程开发人员设置模型测试参数，执行模型测试开发人员根据模型测试时打印的日志观察P、R、F1值和预测时间，了解模型的预测性能

表3.4为模型选择部分的用例描述，在模型训练和测试完毕后，业务人员可以通过模型选择来进行模型组合进行信息抽取。

表3.4 模型选择用例描述

用例编号 UC04

用例名称模型选择

用例描述业务人员选择模型进行信息抽取

参与者业务人员

触发条件业务人员在配置文件中设置模型名称

前置条件模型测试完成

后置条件模型选择完成

优先级高

正常流程业务人员设置配置文件中的模型名称模型选择完成

表3.5为信息抽取服务部分的用例描述，业务人员可以调用信息抽取服务，输入一段非结构化文本，选择模型后，得到最终信息抽取后的结果。

表3.5 信息抽取服务用例描述

用例编号 UC05

用例名称信息抽取服务

用例描述开发人员将模型部署在集群上，暴露端口提供信息抽取服务；业务人员通过访问端口调用信息抽取服务

参与者开发人员，业务人员

触发条件业务人员调用服务

前置条件模型已保存且服务已部署在集群上

后置条件信息抽取完成

优先级高

正常流程用户输入非结构化文本数据系统返回信息抽取结果

3.1.2 非功能性需求

信息抽取服务系统的非功能需求包括响应性能、易用性、可维护性、负载均衡等需求：

1. 响应性能：系统需要通过服务器处理数据，再将数据通过APP返回给用户，如果响应时间过长会导致用户使用体验降低，所以需要保证系统的响应性能。

2. 易用性：系统的易用性主要体现在用户交互友好，提供的功能简单易懂，用户能够轻松理解系统的功能并进行操作使用。

3. 可维护性：系统的可维护性包括系统出错后的可修复性和系统升级时的可修改性。本文实现的信息服务服务系统在每个步骤都有详细的报错机制，系统遇到任何问题都会在日志中留下详细的报错信息，精确定位出错位置，进行修复。系统在要增加新的功能时不用更改之前的代码，在增加新功能时尽可能地避免改动。

4. 负载：由于是真实场景中的信息抽取服务，所以必须至少支持一定数量的用户同时调用信息抽取服务，保证服务的正常运行。

3.2 信息抽取服务总体设计

3.2.1 项目总体规划

图3.2 信息抽取服务系统总体功能模块结构示意图

图3.2所示的为整个系统的功能模块结构示意图，本文所实现的是在构建知识图谱过程中对非结构化数据进行信息抽取的一个信息抽取服务系统。在对非结构化数据信息抽取时通常需要通过数据预处理再进行命名实体识别和关系抽取形成构建知识图谱的三元组数据，所以整个信息抽取服务系统总共分成三大模块，分别是数据预处理模块、信息抽取模型模块和信息抽取服务模块。

其中数据预处理模块包含了必要的数据清洗，将输入的原始数据中各种与任务无关的文本进行过滤，如繁体字转换为简体字、HTML文件中的各种符号等。数据预处理模块还包括在命名实体识别和关系抽取任务中需要用到的词向量模型的训练。

信息抽取模块又分为命名实体识别模块和关系抽取模块两个子模块，两个子模块中分别包括模型训练、模型选择和模型测试。用户可以设置不同的超参数进行模型训练，训练后的模型可以保存并固化成pb文件供后续的任务进行模型选择和调用。在模型选择方面，根据实际需要，命名实体识别模块提供两种不同的模型供用户进行选择，第一种是基于word2vec词向量的Bi-LSTM+CRF模型，第二种是基于BERT的Bi-LSTM+CRF模型。word2vec+Bi-LSTM+CRF模型在模型预测效果上不如BERT+Bi-LSTM+CRF模型，不过预测速度和训练速度更快，所以第一种模型比较适合大批量、对预测精度要求没那么高的情况，第二种模型比较适合小批量、对预测精度要求较高的情况，用户可以根据不同的情况选择不同的模型组合。用户在设置不同超参数对模型训练后，可以利用模型在测试集上的预测效果对模型进行性能评估来测试模型。

信息抽取服务模块是指在服务部署到集群上后，会暴露端口供用户进行访问，用户只需要将需要进行信息抽取的非结构化文本传入即可。之后的一系列数据处理、信息抽取都会在服务端完成，最后返回给用户信息抽取后得到的三元组数据。

3.2.2 系统逻辑视图

图3.3 系统逻辑视图

上一节主要介绍了整个系统的功能模块结构以及每个模块的大致功能，如图3.3所示为信息抽取服务系统的逻辑视图。从整个系统出发，信息抽取服务系统提供给客户端一个服务调用接口来进行信息抽取请求，通过HTTP请求appserver组件，appserve再通过RPC方式调用data_process，通过data_process组件进行数据预处理，数据预处理后model_train组件、model_test组件、model_builder组件分别可以通过data_process组件获取数据，并分别执行train、test、predict操作。

3.2.3 系统开发视图

图3.4 系统开发视图

如图3.4所示为信息抽取服务系统的开发视图，信息抽取服务系统主要有数据预处理模块、信息抽取模型模块和信息抽取服务模块，系统的所有模块都最终部署在Kubernetes集群上，客户端通过HTTP请求方式调用信息抽取服务。数据预处理模块主要负责对输入数据进行清洗、预处理和词向量的训练与保存；信息抽取模型模块中包含了模型训练、模型测试和模型选择；信息抽取服务模块主要负责将模型包装成app并通过服务器向客户端返回模型预测结果。

3.3 信息抽取服务模块设计

3.3.1 数据预处理模块设计

数据预处理模块的功能是对用户的输入数据进行数据清洗，再训练词向量。数据预处理模块的流程图如图3.5所示。

图3.5 数据预处理模块流程图

在数据预处理模块中，首先读取输入的非结构化文本数据，将文本中所有标点符号、空格、回车等进行过滤，得到纯文本数据，再将数据中的繁体字转化为简体字，得到简体中文文本数据。得到简体中文文本数据后，对文本数据进行分词，对分词后的数据进行词向量模型预训练，如果已经有预训练好的词向量，则使用已保存的词向量进行后续模型训练。

3.3.2 信息抽取模型模块设计

信息抽取模型模块包括了命名实体识别模块和关系抽取模块两个子模块。两个子模块中又分别包含模型的构建、模型选择、模型训练和模型测试几部分内容。

图3.6 信息抽取模型模块类图

整个信息抽取模型模块的详细设计类图如图3.6所示。在信息抽取模型模块中，Server_start、LoadData、Model_Builder、Test为两个子模块的公共部分。Server_start为整个模块的入口，从Server_start中进行设置来选择模型，并设置相关参数；在Server_start中选择完模型后，进入LoadData读取从数据预处理模块传入的文本数据和预训练好的词向量；Model_Builder负责对选择好的模型进行初始化，在模型训练完后保存训练好的模型；Test负责读取测试集数据，将训练好的模型在测试集上进行预测，预测完成后返回测试数据；Models部分分为NER_Model和RE_Model，分别提供了2种命名实体识别模型和1种关系抽取模型。NER_Model中的两种模型分别为基于词向量的Bi-LSTM+CRF模型和基于BERT的Bi-LSTM+CRF模型，RE_Model中的模型为ResCNN模型。

3.3.3 信息抽取服务模块设计

图3.7 信息抽取模块设计图

信息抽取服务模块设计图如图3.7所示。在信息抽取模型模块中模型训练和测试完毕后，需要把模型封装并部署到公司集群上。本文的模型部署环境为公司的Kubernetes集群，集群运行的是打包封装了模型、打包了各种依赖并构建了镜像的Docker虚拟机，Docker虚拟机从镜像仓库中调用相关镜像来启动信息抽取服务。本文所述项目实现的信息抽取服务是通过客户端-服务器的架构实现的，客户端的功能是负责接收用户输入的非结构化文本数据，通过服务器暴露的端口进行访问，调用服务器中的信息抽取功能。客户端在整个信息抽取服务系统中只负责与用户进行交互并接收数据，所有的数据处理和分析功能都由服务端来完成，这样做能尽可能地减少服务的复杂度。服务器中的虚拟机需要封装好模型、各种数据处理的接口，接收客户端传来的数据进行数据处理并调用模型进行预测，返回最终结果。

3.4 命名实体识别模型结构设计

图3.8 Bi-LSTM+CRF模型结构示意图

图3.8所示为模型结构示意图。本文在解决命名实体识别任务时选择了两种不同的预训练模型来提取输入文本的语义特征，在网络层都使用Bi-LSTM，在输出时都使用CRF+Softmax进行分类，得到最终的分类标签。本节将分别介绍这两种预训练模型结合Bi-LSTM+CRF网络结构的设计方案。

先把输入文本按字组成一个输入文本的字典，字典中每个字都有一个唯一的ID（类似于唯一标识符），再把输入文本对照字典得到每个字在字典上的相对位置和字向量进行嵌入（embedding）操作，得到每一个字的向量。本文使用的字向量模型有两种：一种是基于金融领域上市公司新闻、金融公告、行业信息等中文语料，使用Gensim库中的word2vec方法训练得到的100维字向量。另一种是Google提出的BERT中文预训练模型，是一个768维的字向量。把输入文本的字向量和词汇特征作为模型的输入，在中间网络层上使用Bi-LSTM，通过前向隐藏状态传递和后向隐藏状态传递分别对输入序列进行编码操作（即进行特征提取），有效地利用了前向和后向的所有信息（即上下文信息），最后利用CRF层从这些特征中计算出序列中每一个元素的标签。

3.4.1 模型的标注策略

由于本文所研究的是金融领域下的信息抽取任务，所以标注语料都是来自于金融领域的新闻文本、上市公司信息、金融公告等。在做金融领域的命名实体识别时，本文将实体的类型分为公司（C）、人（P）、组织机构（INS）、行业（IND）、地点（L）、主营业务（PRO）六大类。本文对中文语料采用的是BIO标注方式，具体格式如图3.9所示。

腾 B-C讯 I-C科 I-C技 I-CC OE OO O马 B-P化 I-P腾 I-P

图3.9 BIO标注格式示例

B-标签1作为命名实体第一个字的标签，如果命名实体的字数大于1，则后面的字的标签都为I-标签1；如果命名实体只有一个字的话，则这个字的标签为B-标签2（中文中一个字作为命名实体的情况较少）；如果文本中的字不为命名实体的组成部分，则标签为O。所以最终得到的标签有：B-C、B-P、B-L、B-IND、B-INS、B-PRO、I-C、I-P、I-L、I-IND、I-INS、I-PRO、O共13种标签类型。这样做的好处是能够把命名实体和非命名实体的字词有效地分隔开来，有利于之后的输出层进行标签分类。

BERT模型由于其自身的结构，在此基础上增加了[CLS]、[SEP]、[X]三个标签类型，BERT的输入是两句话的组合，第一句话的开头用[CLS]标识，每句话的结尾用[SEP]作为标识，而[X]用来表示输入中被遮蔽（Masked）的单词。

3.4.2 基于word2vec的输入层

基于word2vec的模型的输入层是将文本输入的单词序列进行处理转化成单词的特征向量序列进行汇总，而单词的特征向量是由输入文本的字向量和额外特征向量拼接组成的。如图3.10所示。

图3.10 基于word2vec的句子级别命名实体识别模型架构图

由于本文的任务是中文命名实体识别，它属于一项序列标注任务，标注的对象是文本中的每个字，得到的输出也是针对每个字进行一个预测标签的计算，所以本文使用的是字向量。

使用预训练好的字向量能够使输入序列包含了字符的语义信息，所以能够提高模型的效果。图3.11所示为词向量使用示意图。

图3.11 词向量使用示意图

对于输入文本，除了可以用文本的字向量作为其特征向量外，还可以引入额外的特征向量。本文在选取额外的特征向量时，把输入文本的词汇信息作为额外特征。本文采用jieba分词（jieba分词为Pyhton编程环境下的一个分词工具包），并结合金融领域的自定义词典对输入文本进行分词，对分词后的文本进行标记：若单词的字符长度大于等于3，则单词的第一个字符被标记为1，最后一个字符被标记为3，中间的所有字符被标记为2；如果单词的字符长度等于2，则单词的第一个字符被标记为1，第二个字符被标记为2；如果单词的字符长度为1（即1个字），则这个字符被标记为0，如图3.12所示。

seg_feature:北京市海淀区国资委已与江苏银行北京分行签署战略合作协议

[1,2,3,1,2,3,1,2,3,0,0,1,2,2,2,2,2,2,3,1,3,1,2,2,2,2,3]

图3.12 词汇信息示例

最后将输入文本的字向量和分词后的词汇特征向量做拼接后作为模型的输入层结果输入到网络层中进行编码操作。

由于深度学习模型中模型的参数很多，而标注好的训练样本又比较少，所以训练深度学习模型的时候最容易出现的问题就是模型的过拟合问题，模型的过拟合具体表现在：模型在训练数据上的损失函数值较小，预测的准确率较高。但在测试数据上的损失函数值较大，预测的准确率较低。本文在输入层和网络层之间加了Dropout来缓解神经网络过拟合的发生[Hinton et al., 2012]，在一定程度上起到了正则化的作用。

Dropout之所以可以缓解神经网络过拟合的发生，是因为在前向传播时，Dropout让某个神经元的激活值以一定的概率p停止工作，这样可以使得模型的泛化能力变强，因为这样可以让模型不会太依赖某些局部的特征。

3.4.3 基于BERT的输入层

基于BERT的模型是将每个词对应的词向量、句子标签、相对位置向量相加得到的总特征向量作为输入，如图3.13所示。

图3.13 BERT模型输入示例

本文使用的BERT模型是网络层为12层，隐藏层为768层，自注意力头个数为12的网络模型，结合MaskedLM和句子级别的连续性预测任务进行模型的预训练。

BERT模型作为预训练模型在整个命名实体识别模型中的作用同3.3.1节中的word2vec预训练得到的词向量一样，同样作为输入文本的表征传入后续的网络层进行编码，在传入网络层前同样加了Dropout来缓解过拟合。

3.4.4 网络层

一般来说同一个任务的不同模型主要的差异就是网络层选择的编码器不同，本文采用的是Bi-LSTM作为编码器对模型输入的特征向量进行处理，得到每个字符的预测特征向量，再传输给CRF层计算得到每个字符最终的预测标签。

图3.14 基于Bi-LSTM的编码层结构图

如图3.14是一个基于双向长短时记忆模型的编码层的结构图。假设图中维度为n的向量代表当前时刻输入的一个字对应的总特征向量。先将输入到网络层的双向长短时记忆模型中，分别与前向层、后向层中传递过来的t时刻的前、后的状态信息进行计算，得到t时刻前、后时刻的状态对当前状态的影响。再把前向层和后向层的当前时刻的输出做拼接，得到长度为2*n的向量作为网络层的输出。

3.4.5 输出层

输出层的作用是把网络层编码后的输出进行处理后得到最终的每个字符对应的预测标签。本文的输出层是由Softmax全连接层和CRF层联合构成。

图3.15 全连接层示意图

如图3.15所示，在Softmax层中假设输出层得到的网络层输出为，其中为当前时刻的字符对应的模型预测向量，会经过计算得到输入对应的状态分数矩阵。其中W的维度为(标签类别数量, 字向量的维度)，偏置项b的维度为标签类别数量。对于输出层来说需要训练迭代更新的参数就是W和b。

在一般的命名实体识别模型中，即使不加入CRF层，即用Softmax层计算得到的状态分数矩阵并取其中的最大值作为预测标签的依据也可以得到最终的分类标签结果，但是那样的结果会存在一些问题导致最终的分类效果不好，如图3.16所示。

图3.16 未使用CRF层的模型示例

显然图中的分类结果并不准确，这也是为什么需要引入CRF层：CRF层可以学习到句子的约束条件，CRF层在训练数据时可以自动学习到这些约束，保证最终预测结果是有效的。

其中可能的约束条件有：

1. 句子的开头应该是“B-”或者“O”，而不是“I-”。

2. “B-标签1 I-标签2 I-标签3...”这样的模式中，标签1、标签2、标签3应该是同一种实体类别。比如，“B-P I-P I-P”这样的序列是对的，而“B-P I-C I-C”这样的序列是错误的。

有了这些约束条件，模型预测出错的概率将会大大降低。

CRF层接收到全连接层计算得到的状态分数矩阵为SC，代表序列中第i个字符预测为标签j的得分。CRF层中除了状态分数矩阵S，还有一个非常重要的分数---转移分数t。转移分数指的是序列从标签1转移到标签2的分数，如“B-C”转移到“I-C”的分数。为了使转移分数矩阵具有更好的鲁棒性，本文在每个句子的开头加上START标签作为句子开始的标记（所以此时的标签类别个数为14个），在转移分数矩阵中START的分数为0。表3.6为转移分数矩阵的示例。

表3.6 转移分数矩阵示例

	START	B-P	I-P	B-INS	I-INS	O
START	0	0.8	0.007	0.7	0.0008	0.9
B-P	0	0.6	0.9	0.2	0.0006	0.6
I-P	-1	0.5	0.53	0.55	0.0003	0.85
B-INS	0.9	0.45	0.0003	0.25	0.8	0.77
I-INS	-0.9	0.45	0.007	0.7	0.65	0.76
O	0	0.65	0.0007	0.7	0.0008	0.9

从表中可以看出转移矩阵已经学到了一些有用的约束信息，如：从“START”到“I-P”或者“I-C”的分数很低，但到“B-P”和“B-C”的分数很高，可以看出句子的第一个单词的标签应该是“B-”或者“O”，而不是“I-”；“O”后面接“I-标签”的分数很低，可以看出命名实体的开头应该是“B-”而不是“I-”。

设为CRF层的转移分数矩阵，代表从标签i转移到标签j的分数。转移分数矩阵作为Bi-LSTM+CRF模型中的一个参数，在训练模型之前可以随机初始化得到这样一个矩阵T，T中的分数会随着模型训练的更新迭代不断更新，也就是说CRF层可以自己学到这些约束。

假设输入序列X的可能预测序列类别为,每种可能预测序列标签的分数为，共有n条路径，那么路径的总分为：

其中。那么得到正确的预测序列标签的条件概率为

随着模型训练的进行，模型的参数不断更新，使得真实的预测序列标签占所有预测路径的比例越来越大。取的负对数作为损失函数Loss，训练的目标为最小化损失函数以此来调整模型参数，即：。训练完成后，使用viterbi_decode()调用维特比算法计算出概率最高的序列标签作为预测序列标签。

3.5 关系抽取模型结构设计

深度残差网络作为一种比较新的深度神经网络模型常被用于图像领域相关任务，因为它在层数较深时的表现十分优异。本文使用了一种新的CNN模型结合残差学习的方法用于关系抽取问题的研究，不同于以往残差网络的结构较深的特点，本文采用的卷积神经网络结构只有9层，并且得到了不错的分类效果。实验中还发现使用身份映射（identity mapping）能够显著提升远程监督关系抽取问题的性能。

1. 训练好的模型；Test负责读取测试集数据，将训练好的模型在测试集上进行预测，预测

9. 6549072_安磊_基于深度学习的信息抽取服务系统的设计与实现_基于深度学习的信息抽取服务系统的设计与实现.doc_第9部分

总字数：2480

相似文献列表

去除本人已发表文献复制比：7.9%(197)　　文字复制比：7.9%(197)　　疑似剽窃观点：(0)

1	基于灰色神经网络的直升机旋翼故障诊断技术 周鸿灯(导师：高亚东) -《南京航空航天大学硕士论文》- 2018-03-01	6.1%（152） 是否引证：否
2	基于FPGA的目标检测算法加速与实现 吴晋(导师：王东) -《北京交通大学硕士论文》- 2018-06-04	2.1%（52） 是否引证：否
3	基于LSTM融合多CNN的事件图像分类研究 汤华东(导师：侯建军) -《北京交通大学硕士论文》- 2018-05-01	1.7%（41） 是否引证：否
4	010_21401026_李想 李想 -《学术论文联合比对库》- 2017-05-08	1.4%（35） 是否引证：否

原文内容

ResCNN主要的结构分为输入层、卷积层、残差卷积块、最大池化层和输出层几部分。模型的结构图如图3.17所示。

图3.17 ResCNN模型结构示意图

输入层包括词向量层和位置向量层；卷积层是对输入层输出的文本特征向量做卷积操作来提取特征；残差卷积块利用残差学习结合卷积神经网络对卷积层的输出进行进一步处理；最大池化层对残差卷积块的输出进行最大池化操作并取其中的最大值；最后把这些特征传入Softmax层进行计算得到最终的预测结果。

3.5.1 输入层

模型的输入层是将文本输入的单词序列进行处理转化成单词的特征向量序列汇总后传输进后续的卷积层进行卷积处理，而单词的特征向量是由输入文本的词向量和相对位置向量拼接组成的。

假设是输入文本的第i个词，是文本中的实体对，将在预训练好的字向量表中对应的字向量设为，相对位置向量设为，最终把和拼接起来变成词的总特征向量。

由于本节的关系抽取任务是接着上一节的命名实体识别之后继续对金融领域的文本进行实体间关系抽取，所以本节的关系抽取模型使用的字向量也是基于之前的金融领域上市公司新闻、金融公告、行业信息等中文语料，利用Gensim库中的word2vec方法进行预训练得到的字向量模型。

每个对应的总特征向量都是一个实值向量，所有的向量组成了一个特征向量矩阵，其中V是一个输入文本的所有单词构成的词表。

在对输入文本做特征提取时，通常的做法是对输入文本中的单词做一个词向量转换来学习到文本中的语义信息。然而在关系抽取任务中，要想最终获得两个实体间的关系，不仅可以从词汇信息和语义信息的角度出发，还可以从当前单词与实体之间的相对位置中获取一些有用的信息[Zeng et al., 2014]。

相对位置向量结合了当前字符分别与两个实体和的相对距离，比如在句子""中，与和的相对距离分别为1和-3。之后通过一个随机初始化的相对位置矩阵将相对位置距离转化成实值向量。如果当前字符距离命名实体的距离过远，那么可能会被认为与实体间的关系没有关联而在最后分类的时候被忽略。

最后把每个字的字向量和相对位置向量做拼接，再把句子中的每个字的总特征向量拼接成一个向量：

3.5.2 卷积层

卷积层的作用是把输入层输出的总特征向量进行卷积操作后学习到输入文本的潜在特征。

用来表示特征向量拼接得到的向量。卷积操作就是用一个大小为h的窗口在卷积层的输入特征向量上进行“滑动“，对窗口中的所有字符对应的特征向量元素做矩阵计算，最终得到新的输出作为特征C。特征是由一个滑动窗口中的字符对应的特征向量通过公式计算得到的。其中b为偏置项，f是一个非线性函数，。卷积神经网络的窗口大小也就是卷积核的大小，卷积核不断滑动直到遍历整个输入矩阵，将输入通过计算加权和的方式对输入数据做处理。而输入数据是否能进入窗口扫描的区域直接决定了数据经过卷积核的计算后的输出结果，所以卷积核的尺寸直接决定了新的特征C汇聚了多少输入特征。如果是深度神经网络（DNN），那么使用标准的全连接层产生的参数个数就会大大增多，每个输出都是所有输入数据的加权求和得到的结果。但卷积操作的话只允许用个参数来实现这个变换，每个输出特性并不需要“利用”每个输入特征，而只“利用”来自大致相同位置（周边位置）的输入特征。

最终通过滑动窗口使所有的特征向量经过卷积操作得到了特征作为卷积层的输出。

3.5.3 残差卷积块

残差学习将低层的信息直接与高层的信息进行联接来消除深层网络中的梯度消失问题，本文中的残差卷积块就是利用了快捷连接（shortcut connections）对卷积层的输出进行处理。

每个残差卷积块是一个由2层卷积神经网络组成的序列，所有卷积神经网络中的卷积核大小都为h，每层卷积神经网络后还接了一个ReLU激活函数。ReLU激活函数常在训练神经网络时作为中间隐藏层的一种激活函数，其表达式为。相比于最常用的Sigmoid激活函数，由于优化参数时一般会用到误差反向传播算法，要对激活函数进行求导，而Sigmoid激活函数的导数表达式为，Sigmoid激活函数图像如图3.18所示。

图3.18 Sigmoid激活函数

由图可以看出Sigmoid激活函数的导数从0开始后很快就会趋近于0，容易产生“梯度消失”现象，而ReLU激活函数就不会

存在这样的情况，它的导数表达式为，对应的图形如图3.19所示。

图3.19 ReLU激活函数

使用ReLU作为激活函数的好处有：

1. 单侧抑制。即使所有的负值都变为0，而正值的数值不变。正因为ReLU的单侧抑制才使得神经网络中的神经元具有了稀疏激活性，也就是说当模型的层数增加了N倍后，ReLU神经元的激活率将降低2的N次方倍。

2. 对于线性函数而言，ReLU的表达能力更强，尤其在网络很深的情况下。

3. 对于非线性函数而言，ReLU由于非负区间的梯度为常数，所以不存在梯度消失的问题，而且模型的收敛速度维持在一个比较稳定的状态。

设残差卷积块中的2层卷积神经网络的卷积过滤器分别为和，。第一个卷积过滤器（filter）：。第二个卷积过滤器：

其中b1和b2都是偏置项。最后再通过残差学习操作：

本文实现的模型中使用了9个这样的残差卷积块，进行了一系列的卷积+残差连接的操作后最后得到的特征矩阵仍记为C。

3.5.4 最大池化层

对残差卷积块得到的特征c采用最大池化操作，得到每个特征c对应的最大值。由于每个过滤器都可以提取一个特征，所以共提取到m个特征（本文中m为32）。

3.5.5 输出层

模型的输出层是一个全连接的Softmax层，输入为最大池化层中提取出来的m个特征，输出的是通过使用Dropout进行全连接得到的关系的概率分布。

3.6 本章小结

本章主要介绍了信息抽取服务系统的需求分析、服务系统的总体设计，对信息服务系统进行了模块划分，对系统中的每个模块进行详细设计以及信息抽取中两个子任务命名实体识别和关系抽取模型的模型结构进行详细设计。

指　标

疑似剽窃文字表述

1. 正因为ReLU的单侧抑制才使得神经网络中的神经元具有了稀疏激活性，也就是说当模型
2. 对于线性函数而言，ReLU的表达能力更强，尤其在网络很深的情况下。
3. 对于非线性函数而言，ReLU由于非负区间的梯度为常数，所以不存在梯度消失的问题，而且模型的收敛速度维持在一个

10. 6549072_安磊_基于深度学习的信息抽取服务系统的设计与实现_基于深度学习的信息抽取服务系统的设计与实现.doc_第10部分

总字数：15793

相似文献列表

去除本人已发表文献复制比：18.4%(2904)　　文字复制比：18.4%(2904)　　疑似剽窃观点：(0)

1	011_M201570005_聂国平 聂国平 - 《学术论文联合比对库》- 2017-05-18	15.2%（2404） 是否引证：否
2	基于卷积神经网络的中文文本分类研究 聂国平 - 《学术论文联合比对库》- 2017-04-26	15.2%（2404） 是否引证：否
3	论文Convolutional Naural Networks for Sentence Classification--TensorFlow实现篇 - liuchonge的博客 - 博客频道 - CSDN.NET -《网络（http://blog.csdn.net）》- 2017	13.7%（2163） 是否引证：否
4	Tensorflow实现的CNN文本分类 - somTian的博客 - 博客频道 - CSDN.NET -《网络（http://blog.csdn.net）》- 2017	11.6%（1827） 是否引证：否
5	基于tensorflow的cnn文本分类 - u013713117的专栏 - CSDN博客 -《网络（http://blog.csdn.net）》- 2017	11.2%（1772） 是否引证：否
6	Tensorflow版TextCNN主要代码解析 - u013818406的博客 - CSDN博客 -《网络（http://blog.csdn.net）》- 2017	11.1%（1752） 是否引证：否
7	CNN情感分析（文本分类）- 萝卜虫的博客 - CSDN博客 -《网络（http://blog.csdn.net）》- 2017	9.6%（1519） 是否引证：否
8	卷积神经网络实践 - Abrohambaby的博客 - CSDN博客 -《网络（http://blog.csdn.net）》- 2017	6.9%（1084） 是否引证：否
9	基于深度学习的评论文本情感分类系统设计与实现 周全(导师：维尼拉·木沙江;吐尔地·托合提) -《新疆大学硕士论文》- 2018-06-01	5.3%（843） 是否引证：否
10	基于集成算法的密级文本分类系统设计 顾凯文(导师：孙国梓) -《南京邮电大学硕士论文》- 2018-11-14	5.2%（828） 是否引证：否
11	1049721403027_鞠亮_学术性硕士_信息与通信工程_周祖德 鞠亮 -《学术论文联合比对库》- 2017-06-07	4.9%（772） 是否引证：否

12	基于关键词自学习的中文网页分类技术研究与实现 鞠亮(导师：周祖德) -《武汉理工大学硕士论文》- 2017-03-01	4.9%（770） 是否引证：否
13	纠错： 深度学习模型优化时快速收敛 - Sherry -《网络（http://blog.csdn.net）》- 2017	3.4%（536） 是否引证：否
14	020+1642+刘恒旺 刘恒旺 -《学术论文联合比对库》- 2017-12-18	1.1%（170） 是否引证：否
15	经管院-115107000850-刘恒旺-吴鹏nm 经管院 -《学术论文联合比对库》- 2017-12-27	1.0%（160） 是否引证：否
16	Hadoop分布式集群的自动化容器部署研究 李杰;刘广钟; -《计算机应用研究》- 2016-01-08 1	0.7%（118） 是否引证：否
17	2014262593_尹羿_基于OSGi的政务云组件运行环境研究与实现_软件工程_武君胜 尹羿 -《学术论文联合比对库》- 2017-02-24	0.5%（77） 是否引证：否
18	荣科科技今年跌逾24% 实控人承诺兜底增持 证券时报记者 康殷 -《证券时报》- 2018-09-14	0.4%（67） 是否引证：否
19	docker 标签 -《网络（http://blog.51cto.co）》- 2016	0.4%（62） 是否引证：否
20	煤炭板块利润连续下滑 重庆能源欲转型“卖”咖啡	每经网 -《网络（http://www.nbd.com.c）》- 2016	0.4%（56） 是否引证：否

原文内容

第四章信息抽取服务的实现

第三章中详细阐述了信息抽取服务系统的需求分析、总体规划和各个模块的详细设计。本章在各个模块的详细设计的基础上，对每个模块的具体实现进行了详细的介绍并给出了部分实现代码。

4.1 数据预处理模块的实现

4.1.1 数据采集

本文涉及的信息抽取模型主要是针对金融上市公司相关领域，实验数据来自于金融界、新浪财经、东方财富、每日经济新闻等网站的上市公司的新闻文本、公司公告、最新资讯等文本信息，而采集这些数据的方式是数据爬虫。

经过Scrapy爬虫、数据清洗、人工数据标注后最后得到的命名实体识别数据和关系抽取数据分别如图4.1和4.2所示。

{"entities": [{"word": "重庆能源", "start": 23, "end": 27, "type": "C"}, {"word": "煤炭", "start": 35, "end": 37, "type": "IND"}], "text": "《每日经济新闻》记者注意到，上述评级报告在阐述重庆能源的关注点时提及***“煤炭板块盈利能力较弱，拖累公司经营性业务利润”***"}

图4.1 命名实体识别标注数据示例

金力泰吴国政股东金力泰公告，12月13日，公司控股股东吴国政先生协议转让给柯桥领英的公司47034000股无限售条件流通股（占公司总股本10.00%）已完成了过户登记手续金力泰。

图4.2 关系抽取标注数据示例

由于关系抽取任务的人工标注相对于命名实体识别任务的人工标注而言更加困难和耗时，所以本文采用了远程监督的方法来扩充人工标注的关系抽取语料。经过人工筛选，共选出33种关系作为关系抽取模型的关系标签，表4.1为关系标签列表。

表4.1 关系标签列表

unknown 0实际控制人 1主营业务 2子公司 3合作伙伴 4一致行动人 5收购 6调查 7处罚 8法人 9行业 10关联关系 11高管 12供应商 13监事 14总统 15重组 16下属公司 17附属公司 18夫妻 19执行事务合伙人 20股东 21股权转让 22质押 23设立 24创始人 25董事 26成立 27地址 28曾用名 29CEO 30职工 31亲属 32并购 33

4.1.2 数据预处理

在数据预处理模块，需要对输入的文本数据进行不可用信息过滤包括各种符号、空格、回车符等，中文的繁体字转简体字，将数据处理成训练模型所需的数据结构。

中文的繁简体转换可以通过OpenCC工具，在命令行对数据执行opencc命令即可，如图4.3所示。

opencc -i input.txt -o output.txt -c zht2zhs.ini

图4.3 中文繁简体转换命令

在将输入出具处理成训练模型所需的数据结构时，同时把数据按batch_size的大小进行切分，这里自定义了一个DataManager类，如图4.4所示。

class BatchManager(object): def __init__(self, data, batch_size): self.batch_data = self.sort_and_pad(data, batch_size) self.len_data = len(self.batch_data) def sort_and_pad(self, data, batch_size): num_batch = int(math.ceil(len(data) / batch_size)) sorted_data = sorted(data, key=lambda x: len(x[0])) batch_data = [] for i in range(num_batch): a = int(i*batch_size) b = int((i+1)*batch_size) batch_data.append(self.pad_data(sorted_data[a:b])) return batch_data def pad_data(data): strings = [] chars = [] segs = [] targets = [] max_length = max([len(sentence[0]) for sentence in data]) for line in data: string, char, seg, target = line padding = [0] * (max_length - len(string)) strings.append(string + padding) chars.append(char + padding) segs.append(seg + padding) targets.append(target + padding) return [strings, chars, segs, targets] def iter_batch(self, shuffle=False): if shuffle: random.shuffle(self.batch_data) for idx in range(self.len_data): yield self.batch_data[idx]

图4.4 BatchManager类实现

4.1.3 词向量的训练

在数据处理完成后，需要先预训练词向量。本文使用的词向量有两种：第一种是调用Gensim库中的Word2vec方法，利用爬取的金融领域相关语料进行训练得到的word2vec词向量；第二种是Google开源的，利用维基百科中的中文百科数据进行预训练的BERT模型。由于BERT模型是已经训练好的，所以本文实现的词向量训练是基于Gensim库的word2vec词向量训练，具体实现方法如图4.5，其中参数sentence为数据预处理后的语料，size为词向量的维度大小，window定义了当前词与预测词的最大距离，min_count为一个词最低出现次数，如果小于min_count，则被舍弃。

from gensim.models import word2vecsentence = word2vec.Text8Corpus('./data/output.txt')model = word2vec.Word2vec(sentence, size=100, window=5, min_count=5)model.save('./data/w2v')

图4.5 word2vec词向量实现

4.2 基于word2vec的命名实体识别模型的实现

基于word2vec的命名实体识别模型的主要超参数设置如表4.2所示。其中词向量维度为训练词向量时设置的size大小；词汇特征维度表示词汇特征向量的size大小；LSTM隐藏层大小为LSTM中隐藏节点的数量；batch_size为训练时单次训练用到的数据量；学习率为优化器的学习速度；Dropout比例为Dropout层在选择丢弃时丢弃的比例；优化函数为模型选择的优化器种类；clip的值用来限制梯度的范围。

表4.2 基于word2vec的命名实体识别模型超参数列表

参数名称参数大小

词向量维度 100

词汇特征维度 20

LSTM隐藏层大小 100

batch_size大小 64

学习率 0.001

Dropout比例 0.5

优化函数 Adam

clip大小 5

模型在得到预训练的词向量后，将输入文本转化为词表和标签列表并存入maps.pkl中供模型训练时读取，实现代码如图4.6所示。

if not os.path.isfile(FLAGS.map_file): # create dictionary for word if FLAGS.pre_emb: dico_chars_train = char_mapping(train_sentences, FLAGS.lower)[0] dico_chars, char_to_id, id_to_char = augment_with_pretrained(dico_chars_train.copy(), FLAGS.emb_file, list(itertools.chain.from_iterable([[w[0] for w in s] for s in test_sentences]))) else: _c, char_to_id, id_to_char = char_mapping(train_sentences, FLAGS.lower) # Create a dictionary and a mapping for tags _t, tag_to_id, id_to_tag = tag_mapping(train_sentences) with open(FLAGS.map_file, "wb") as f: pickle.dump([char_to_id, id_to_char, tag_to_id, id_to_tag], f)else: with open(FLAGS.map_file, "rb") as f: char_to_id, id_to_char, tag_to_id, id_to_tag = pickle.load(f)

图4.6 词表和标签列表的存储实现

在构建了词表和标签列表，读取了输入数据后，就可以进行模型的训练：建立模型、设置步长、训练模型、打印日志等，具体实现如图4.7所示。

tf_config = tf.ConfigProto()tf_config.gpu_options.allow_growth = Truesteps_per_epoch = train_manager.len_datawith tf.Session(config=tf_config) as sess: model = create_model(sess, Model, FLAGS.ckpt_path, load_word2vec, config, id_to_char, logger) logger.info("start training") loss = [] for i in range(100): for batch in train_manager.iter_batch(shuffle=True): step, batch_loss = model.run_step(sess, True, batch) loss.append(batch_loss) if step % FLAGS.steps_check == 0: iteration = step // steps_per_epoch + 1 logger.info("iteration:{} step:{}/{}, " "NER loss:{:>9.6f}".format(iteration, step%steps_per_epoch, steps_per_epoch, np.mean(loss))) loss = [] best = evaluate(sess, model, "dev", dev_manager, id_to_tag, logger) if best: save_model(sess, model, FLAGS.ckpt_path, logger) evaluate(sess, model, "test", test_manager, id_to_tag, logger)

图4.7 基于word2vec的命名实体识别模型训练代码

其中create_model()方法构建了整个网络模型，模型结构的具体实现如图4.8所示。

embeddings for chinese character and segmentation representationembedding = self.embedding_layer(self.char_inputs, self.seg_inputs, config)# apply dropout before feed to lstm layerlstm_inputs = tf.nn.dropout(embedding, self.dropout)# bi-directional lstm layerlstm_outputs = self.Bi-LSTM_layer(lstm_inputs, self.lstm_dim, self.lengths)# logits for tagsself.logits = self.project_layer(lstm_outputs)# loss of the modelself.loss = self.loss_layer(self.logits, self.lengths)with tf.variable_scope("optimizer"): self.opt = tf.train.AdamOptimizer(self.lr) # apply grad clip to avoid gradient explosion grads_vars = self.opt.compute_gradients(self.loss) capped_grads_vars = [[tf.clip_by_value(g, -self.config["clip"], self.config["clip"]), v]for g, v in grads_vars] self.train_op = self.opt.apply_gradients(capped_grads_vars, self.global_step)# saver of the modelself.saver = tf.train.Saver(tf.global_variables(), max_to_keep=5)

图4.8 基于word2vec的命名实体识别模型结构实现

输入层将输入单词序列和文本的额外特征做嵌入操作，得到输入文本的特征矩阵表示。在输入层到网络层中间使用tf.nn.dropout()添加Dropout层来缓解过拟合。网络层中分别定义了前向和后向相连的LSTM网络，也就是Bi-LSTM，对输入的上下文信息进行编码。输出层中的全连接层和CRF层在分别调用tf.nn.xw_plus_b()进行矩阵乘法以及偏置项的加法和crf_log_likelihood()计算损失后，选用Adam优化函数并设置clip来限制梯度大小，最后再利用tf.train.Saver()保存模型。

4.3 基于BERT的命名实体识别模型的实现

基于BERT的命名实体识别模型的主要超参数设置如表4.3所示。其中隐藏层大小为BERT预训练模型的网络层数

；Dropout比例为Dropout层选择丢弃输入时的丢弃比例；文本最大输入长度为输入文本的最大长度，如果超过这个长度，超过的部分将被截去并忽略；自注意力头个数为BERT模型中self-attention head的数量；LSTM隐藏节点数量是在网络层中LSTM的隐藏节点数；学习率为优化器的学习速率；batch_size为单次训练时传入的数据量；clip用来将梯度限制在一个阈值内。

·表4.3 基于BERT的命名实体识别模型超参数列表

参数名称参数大小

隐藏层大小 12

Dropout比例 0.01

文本最大输入长度 512

自注意力头个数 12

LSTM隐藏节点数量 768

学习率 0.005

batch_size 32

clip大小 5

由于本文中的基于BERT的命名实体识别模型是指利用已经训练好的BERT模型结合Bi-LSTM+CRF在命名实体识别任务上进行微调，得到最终的实体识别分类。BERT的任务是利用自定义的处理器（DataProcessor）对输入文本进行处理，通过BERT模型得到输入文本的表征，再传入至网络层进行编码，最后进行预测。其中自定义处理器以及通过BERT得到输入文本表征的具体实现如图4.9和图4.10所示。

class NerProcessor(DataProcessor): def get_train_examples(self, data_dir): return self._create_example(self._read_data(os.path.join(data_dir, "train.txt")), "train") def get_dev_examples(self, data_dir): return self._create_example(self._read_data(os.path.join(data_dir, "dev.txt")), "dev") def get_test_examples(self, data_dir): return self._create_example(self._read_data(os.path.join(data_dir, "test.txt")), "test") def get_labels(self): return ["O", "B-P", "I-P", "B-C", "I-C", "B-L", "I-L", "B-INS", "I-INS","B-IND", "I-IND","B-PRO", "I-PRO","X", "[CLS]", "[SEP]"] def _create_example(self, lines, set_type): examples = [] for (i, line) in enumerate(lines): guid = "%s-%s" % (set_type, i) text = tokenization.convert_to_unicode(line[1]) label = tokenization.convert_to_unicode(line[0]) if i == 0: print(label) examples.append(InputExample(guid=guid, text=text, label=label)) return examples

图4.9 自定义处理器NerProcessor的实现

def create_model(bert_config, is_training, input_ids, input_mask, segment_ids, labels, num_labels, use_one_hot_embeddings): # 使用数据加载BertModel,获取对应的字embedding model = modeling.BertModel(config=bert_config, is_training=is_training, input_ids=input_ids, input_mask=input_mask, token_type_ids=segment_ids, use_one_hot_embeddings=use_one_hot_embeddings) # 获取对应的embedding 输入数据[batch_size, seq_length, embedding_size] embedding = model.get_sequence_output()

图4.10 BERT模型加载和输入文本表征获取的实现

通过BERT得到输入文本的表征后，将文本表征作为输入传入到Bi-LSTM+CRF网络模型中进行命名实体识别，这里的模型结构与基于word2vec的命名实体识别模型相同，只在超参数的设置上有所区别。

4.4 关系抽取模型的实现

关系抽取模型的主要超参数设置如表4.4所示。卷积核的大小决定了在卷积操作中滑动窗口每次“滑过”的特征数量；字向量用的是金融领域信息抽取任务的预训练字向量；过滤器数量表明可以对输入信息进行多少种特征计算；学习率表明优化器的学习速率；最大最小相对位置为相对于实体的位置；batch_size、Dropout比例的定义都与前文中的含义相同，不予阐述。

表4.4 关系抽取模型超参数列表

参数名称参数大小

卷积核大小(filter_size) 3

字向量维度 100

相对位置向量维度 5

最大相对位置 50

最小相对位置 50

过滤器数量 32

batch_size 32

学习率 0.001

Dropout比例 0.5

对于关系抽取模型，模型主要的训练流程代码如图4.11所示。先使用tf.ConfigProto()对session进行参数配置，再构建ResCnn模型，设置Adam优化器和学习率并利用计算得到的loss对网络中的参数进行更新。

with tf.Graph().as_default(): session_conf = tf.ConfigProto(allow_soft_placement=FLAGS.allow_soft_placement, log_device_placement=FLAGS.log_device_placement) sess = tf.Session(config=session_conf) with sess.as_default(): cnn = ResCNN(FLAGS.sequence_length, len(datamanager.relations), FLAGS.embedding_dim, 5, list(map(int, FLAGS.filter_sizes.split(","))), FLAGS.num_filters, FLAGS.l2_reg_lambda) # Define Training procedure global_step = tf.Variable(0, name="global_step", trainable=False) optimizer = tf.train.AdamOptimizer(1e-3) grads_and_vars = optimizer.compute_gradients(cnn.loss) train_op = optimizer.apply_gradients(grads_and_vars, global_step=global_step) saver = tf.train.Saver(tf.global_variables()) sess.run(tf.global_variables_initializer())

图4.11 关系抽取模型训练流程代码

ResCNN的网络层代码如图4.12所示。通过tf.nn.conv2d()实现卷积层，其中卷积块的尺寸为[filter_size,

embedding_size+2*position_size, 1, num_filters]，分别是卷积核的大小、卷积核的维度、通道数和卷积核的数量；strides表示在4个维度上的卷积块的滑动步长都为1；padding参数设置为“VALID”表示卷积核在滑动时不可以停留在向量的边缘。之后调用tf.nn.relu()引入了一个ReLU非线性激活函数来防止之后的反向传播出现梯度爆炸和梯度消失。然后将卷积层通过激活函数的feature map和通过残差卷积块Cnnblock得到的feature map相加再传入到最大池化层。最大池化层调用tf.nn.max_pool()方法实现，并将最后的结果通过tf.concat()进行拼接，再调用tf.reshape()将特征矩阵的维度转换成[-1, num_filter_total]得到最终的隐藏层特征矩阵，再在后面接一层Dropout层以防过拟合。

pooled_outputs = [] for i, filter_size in enumerate(filter_sizes): with tf.name_scope("conv-maxpool-%s" % filter_size): # Convolution Layer filter_shape = [filter_size, embedding_size+2*position_size, 1, num_filters] W = tf.Variable(tf.truncated_normal(filter_shape, stddev=0.1), name="W") b = tf.Variable(tf.constant(0.1, shape=[num_filters]), name="b") conv = tf.nn.conv2d(self.embedded_chars_expanded, W, strides=[1, 1, 1, 1], padding="VALID", name="conv") # Apply non-linearity, shape of h is [batch_size, sequence_length-2, 1, num_filters] h = tf.nn.relu(tf.nn.bias_add(conv, b), name="relu") self.l2_loss += tf.nn.l2_loss(W) self.l2_loss += tf.nn.l2_loss(b) # nine convolution layers for i in range(3): h2 = self.Cnnblock(num_filters, h, i) h = h2+h pooled = tf.nn.max_pool(h, ksize=[1, sequence_length - filter_size + 1, 1, 1], strides=[1, 1, 1, 1], padding='VALID', name="pool") pooled_outputs.append(pooled) num_filters_total = num_filters * len(filter_sizes) self.h_pool = tf.concat(pooled_outputs, 3) self.h_pool_flat = tf.reshape(self.h_pool, [-1, num_filters_total], name="hidden_feature")with tf.name_scope("dropout"): self.h_drop = tf.nn.dropout(self.h_pool_flat, self.dropout_keep_prob)

图4.12 ResCNN网络层的实现

经过网络层处理后的数据传入给分类器，图4.13为分类器的实现。分类器中的两个主要参数权重和偏置项分别用tf.get_variable()和tf.Variable()进行定义。之后调用tf.nn.xw_plus_b()对网络层输出的特征矩阵进行矩阵乘法并添加偏置项，再调用交叉熵函数，对其求平均值后加上l2正则项得到最后的损失值。

with tf.name_scope("output"): W = tf.get_variable("W", shape=[num_filters_total, num_classes], initializer=tf.contrib.layers.xavier_initializer()) b = tf.Variable(tf.constant(0.1, shape=[num_classes]), name="b") self.l2_loss += tf.nn.l2_loss(W) self.l2_loss += tf.nn.l2_loss(b) self.scores = tf.nn.xw_plus_b(self.h_dropout, W, b, name="scores") self.predictions = tf.argmax(self.scores, 1, name="predictions") with tf.name_scope("loss"): losses = tf.nn.softmax_cross_entropy_with_logits(logits=self.scores, labels=self.input_y) self.loss = tf.reduce_mean(losses) + l2_reg_lambda * self.l2_loss with tf.name_scope("accuracy"): correct_predictions = tf.equal(self.predictions, tf.argmax(self.input_y, 1)) self.accuracy = tf.reduce_mean(tf.cast(correct_predictions, "float"), name="accuracy")

图4.13 关系抽取模型分类器的实现

4.5 信息抽取服务的实现与部署

4.5.1 模型格式转换

在信息抽取模型模块中通过模型训练得到的模型都是checkpoint格式的模型文件，这种格式的文件没有办法直接被读取进行预测，还需要构建完整的模型结构才能读取数据并进行预测。所以为了能够更加方便地在信息抽取服务中提供模型预测功能，需要将模型训练保存下来的checkpoint格式模型文件进行格式转换。在Tensorflow中pb格式的模型文件表示MetaGraph的Protocol Buffer格式文件，MetaGraph是包括计算图，数据流，以及相关的变量和输入输出signature等创建计算图的文件，所以pb格式的模型文件可以直接被调用进行信息抽取服务。图4.14为ckpt格式转pb格式代码实现。

def model_save_pb(ckpt_path, model_meta_name, pbmodel_path, pb_model_name, model_output_nodes): with tf.Session() as sess: sess.run(tf.global_variables_initializer()) latest_ckpt = tf.train.latest_checkpoint(ckpt_path) restore_saver = tf.train.import_meta_graph(os.path.join(ckpt_path,model_meta_name)) restore_saver.restore(sess, latest_ckpt) output_graph_def = tf.graph_util.convert_variables_to_constants(sess, sess.graph_def, [model_output_nodes]) with tf.gfile.GFile(os.path.join(pbmodel_path, pb_model_name), "wb") as f: f.write(output_graph_def.SerializeToString())

图4.14 ckpt格式转pb格式的实现

在会话开启后，调用tf.train.latest_checkpoint()来获取目标路径下最后一次保存的ckpt格式模型文件。为了不重复定义计算图上的运算，调用tf.train.import_meta_graph()直接加载已经持久化的图，再调用restore()加载变量。之后调用tf.graph_util.convert_variables_to_constants()可以将计算图中的变量以常量的形式保存，并且导出了GraphDef部分，GraogDef部分保存了从输入层到输出层的计算过程，在保存的时候保存的是节点名称而不是张量名称。最后得到的pb格式文件中的模型大小会大大减小，非常适合运行和部署。

4.5.2 构建Docker镜像

在编写好数据预处理模块的代码、构建好模型结构并训练完毕转存为pb格式后，就可以将所有的模块和依赖都封装到一个Docker镜像中方便后面的部署。本文的Docker镜像的构建使用的是Dockerfile文件，Dockerfile是一种可以被Docker程序解释的脚本，由一条条指令组成，每条指令对应Linux下面的一条命令，Docker程序解决这些命令间的依赖关系，根据指令生成定制的镜像。当有额外需求需要重新定制镜像时，只需要在Dockerfile中增加或者修改指令并运行，就可以重新生成镜像。本文的Dockerfile内容如图4.15所示。

from 172.16.1.41:5000/modelserver_base:0.6run mkdir /workspacerun mkdir /modelscopy ./models/ /models/copy ./demos/ /workspace/demos/copy dist/ModelServer-1.0.0-cp27-cp27mu-linux_x86_64.whl /workspace/run pip install /workspace/ModelServer-1.0.0-cp27-cp27mu-linux_x86_64.whlworkdir /workspace/demoscmd python /workspace/demos/server_start.py /workspace/demos/config.ini /workspace/demos/register.ini

图4.15 利用Dockerfile构建镜像

其中172.16.1.41:5000/modelserver_base:0.6是一个基础镜像，里面安装了一些项目相关的依赖。之后是创建和复制文件夹，安装压缩包，再使用workdir指定工作目录，最后指定Docker启动镜像时要启动服务py脚本和配置文件。

在编写完Dockerfile后，分别使用docker build和docker run命令制作镜像并启动容器，如图4.16所示。命令中端口映射前面是主机端口，后面是容器环境端口，只有进行绑定才能将主机端口的信息送达到容器的端口，容器才能从自己的端口处得到

信息。

docker build -t information_extraction -f /home/nlp/modelserver .docker run --name nlp-modelserver -p 8080:7790 -d information_extraction

图4.16 制作镜像和启动容器命令

4.5.3 部署到Kubernetes集群

将Docker镜像部署到Kubernetes集群上需要分别构建服务器和客户端两部分，服务器向客户端提供服务，业务人员通过客户端访问服务。构建服务器和客户端分别都需要一个pod和service。服务器的pod用来封装模型、数据、代码等，service用来暴露端口供客户端访问。客户端的pod用来连接服务器并运行服务，service用来暴露端口供业务人员访问并输入数据。

containers:- name:nlp-modelserverimage:information_extractioncommand:-/bin/bash--cargs:--server_host=172.16.30.3--server_port=27852--port=7790--model_name=ner_model--model_base_path=/models/

图4.17 服务器pod的yaml文件部分内容

服务器pod的yaml文件中设置了容器名称、docker镜像名称、args中指定了端口号、模型名称、模型所在路径等。service的yaml文件中指定了容器端口到服务器端口的映射，之后使用kubectl create命令进行创建。

客户端的用于创建pod的yaml文件中指定了执行信息抽取的容器、服务器地址、服务器端口号、客户端端口号、模型名称、模型路径等参数。创建service的yaml文件主要指定了客户端的端口映射。

最后可以通过kubectl get pods命令查看pod是否创建并运行成功。如图4.18所示。

NAMEREADYSTATUSRESTARTSAGEnlp-modelserver1/1Running01m

图4.18 查看pod运行状态

除了观察pod的运行状态，也可以直接通过命令行查看nlp-modelserver的服务情况,如图4.19所示为业务人员选择基于word2vec的命名实体识别模型并启动服务时的部分状态内容。

图4.19 服务启动的状态（部分）

到此，整个信息抽取服务系统基本构建完成。开发人员可以通过系统训练和测试已经构建好的模型，业务人员就可以访问端口进行信息抽取服务的调用，图4.20为公司的人工智能平台通过api请求调用信息抽取服务返回的命名实体识别的结果，可以看到输入文本中的实体和实体类别都被正确预测、返回并显示。如输入为“***9月9日美芝股份公告，控股股东、实际控制人、董事长李苏华向全体员工发出增持公司股票倡议书，鼓励公司及全资子公司全体员工积极买入公司股票。***”，在调用信息抽取服务后返回的结果为标签为company的实体为美芝股份，标签为person的实体为李苏华，结果符合预期。

图4.20 命名实体识别服务返回结果示例

4.6 本章小结

本章主要阐述了信息抽取服务系统中各个模块的具体功能实现。首先是数据预处理模块，包括数据采集和数据处理。之后是两种命名实体识别模型和关系抽取模型的实现。最后是信息抽取服务的搭建与部署。

指　标

疑似剽窃文字表述

1. Dockerfile文件，Dockerfile是一种可以被Docker程序解释的脚本，由一条条指令组成，每条指令对应Linux下面的一条命令，Docker程序解决这些
2. 额外需求需要重新定制镜像时，只需要在Dockerfile中增加或者修改指令并运行，

11. 6549072_安磊_基于深度学习的信息抽取服务系统的设计与实现_基于深度学习的信息抽取服务系统的设计与实现.doc_第11部分　总字数：3437

相似文献列表

去除本人已发表文献复制比：0%(0)　文字复制比：0%(0)　疑似剽窃观点：(0)

原文内容

第五章实验对比与分析

5.1 实验评估标准

本文的实验中采用准确率（Precision）、召回率（Recall）和F1值作为模型的评价指标。其中准确率是指模型预测正确的实体个数占模型预测的所有实体个数的比例，而召回率是指模型预测正确的实体个数占样本中含有的实体总数的比例，FI值则是准确率和召回率的调和平均数：

5.2 命名实体识别模型实验分析

由于本文使用了两种命名实体识别模型，所以将从两种模型的参数设置和不同模型间的性能进行对比实验。

5.2.1 模型参数实验对比与分析

基于word2vec的命名实体识别模型的超参数设置在图4.2中已经给出，经过测试发现，LSTM隐藏层的大小、学习率大小对模型最终的测试结果影响相对较大，表5.1为不同数值下的LSTM隐藏层(h)和学习率(lr)对应的模型的测试结果。针对2个参数的不同取值进行排列组合后，选择模型中实体数量相对平衡的三个类别“公司C”、“人名P”、“组织机构INS”进行测试。对比测试结果可以发现，当LSTM隐藏层大小为100，学习率为0.001时，训练出来的模型测试效果最好。

表5.1 不同超参数下基于w2v的命名实体识别模型测试结果

参数设置 C（%） P（%） INS（%）

P R F1 P R F1 P R F1

h=50,lr=0.001 93.49 94.85 94.16 95.61 96.08 95.84 75.25 73.79 74.51
h=50,lr=0.005 91.68 88.26 89.94 90.33 87.65 88.97 70.28 69.56 69.92
h=50,lr=0.01 89.37 86.25 87.78 90.45 89.86 90.15 70.83 68.67 69.73
h=100,lr=0.001 90.62 86.72 88,63 89.81 85.42 87.56 73.32 70.51 71.89
h=100,lr=0.005 88.86 83.57 86.13 86.28 83.36 84.79 69.98 67.57 68.75
h=100,lr=0.01 87.35 82.19 84.69 86.45 83.86 85.14 68.72 65.43 67.03

5.2.2 不同模型间的实验对比与分析

针对本章需要解决的命名实体识别任务，为了对本文选择的模型进行更全面、更准确的性能评估，本节将列出在相同的输入和语料环境下，CRF、LSTM、Bi-LSTM等模型解决命名实体识别任务时的性能对比，如表5.2所示。

表5.2 不同命名实体识别模型在相同输入下的性能对比

模型名称 C（%） P（%） INS（%）
P R F1 P R F1 P R F1
Bi-LSTM+CRF 93.49 94.85 94.16 95.61 96.08 95.84 75.25 73.79 74.51
CRF 83.26 82.35 82.80 84.19 83.24 83.71 66.25 63.32 64.75
LSTM 84.12 82.62 83.36 85.32 83.37 84.33 66.28 64.58 65.42
Bi-LSTM 87.18 85.27 86.21 88.93 86.54 87.72 69.93 67.72 68.81
BERT+Bi-LSTM+CRF 96.36 96.13 96.23 97.85 97.03 97.44 82.86 80.68 81.76

CRF的分数不高的原因可能是由于CRF无法学习到较长文本中的关联信息；LSTM只学习到了当前词之前的文本信息，没有考虑到下文对当前词的影响； Bi-LSTM没有CRF层来学习约束条件，所以整体效果不如本文所选的Bi-LSTM+CRF模型。而在将word2vec替换成BERT后，模型在各个类别上的预测效果都有了明显的提升，这也证明了BERT相对于word2vec更好地提取了输入文本的语义信息，从而得到了更好的效果。表5.3为基于word2vec的NER模型和基于BERT的NER模型在CPU和GPU环境上对测试集进行预测的速度对比。

表5.3 两种命名实体识别模型在不同环境下预测速度对比

模型名称测试环境测试集大小预测时长
word2vec-NER CPU 22381条 5min
BERT-NER CPU 22381条 76min
word2vec-NER GPU 22381条 24min
BERT-NER GPU 22381条 49.5min

可以看出虽然基于BERT的命名实体识别模型在预测的准确率上效果更好，但是在预测速度上相比于基于word2vec的命名实体识别模型要差10倍不止，所以基于BERT 的命名实体识别模型更适合在小批量、预测精度要求较高的情况下使用，而基于word2vec的命名实体识别模型更适合在大批量、预测精度要求没那么高的情况下使用。此外，基于word2vec的命名实体识别模型在GPU上的预测速度比CPU环境下的预测速度更慢，可能是因为GPU更适合做并行计算，而基于word2vec的命名实体识别模型的预处理逻辑涉及很多顺序计算，无法充分利用GPU的计算能力，这时反而是CPU更加适合。

5.3 关系抽取模型实验分析

5.3.1 模型参数实验对比与分析

在关系抽取模型ResCNN的超参数中，卷积核的大小决定了在卷积操作中滑动窗口每次“滑过”的特征数量，也就等于决定了一次可以对多少个特征进行计算；过滤器数量表明可以对输入信息进行多少种特征计算；学习率可以控制模型的学习进度，如果学习率过大容易震荡且损失值很大，如果学习率过小容易产生过拟合且收敛速度太慢。本文主要通过这几个参数的对比实验来确认不同参数对模型效果的影响，实验采用F1值作为模型性能的评价标准，实验结果如表5.4所示。

表5.4 不同超参数下关系抽取模型测试结果

参数设置 F1值
卷积核大小=3，过滤器数量=50，学习率=0.001 69.24
卷积核大小=3，过滤器数量=50，学习率=0.005 68.96
卷积核大小=3，过滤器数量=50，学习率=0.01 66.32
卷积核大小=5，过滤器数量=50，学习率=0.001 66.07
卷积核大小=5，过滤器数量=50，学习率=0.005 64.23
卷积核大小=5，过滤器数量=50，学习率=0.01 62.18
卷积核大小=7，过滤器数量=50，学习率=0.001 61.34
卷积核大小=7，过滤器数量=50，学习率=0.005 59.35
卷积核大小=7，过滤器数量=50，学习率=0.01 58.27
卷积核大小=3，过滤器数量=100，学习率=0.001 63.45
卷积核大小=3，过滤器数量=200，学习率=0.001 63.26

根据前9行实验对比可以发现，当卷积核大小为3，学习率为0.001时模型的表现效果更好，所以最后2行直接使卷积核大小为3、学习率为0.001来对比过滤器的数量。最后可以发现当过滤器数量为50的时候，模型的表现效果更好，最终得到了表4.4中的超参数设置。

5.3.2 不同模型间的实验对比与分析

本文除了对比了ResCNN中不同的超参数对模型效果的影响，在本章节还对比了ResCNN外别的模型在相同输入和条件下的性能进行模型评估，其它的模型包括：CNN、CNN+Attention，实验的评价标准为模型的F1值，实验结果如表5.5所示。

表5.5 不同关系抽取模型在相同输入下的性能对比

模型名称 F1值
ResCNN 69.24

CNN 53.37

CNN+Attention 65.48

可以看出，由于关系类别较多以及语料数量较少的原因，模型的表现效果都不是特别好，不过相较于CNN和CNN+Attention，ResCNN的性能更加优秀。随着人工标注数据的不断增多并结合一些人为制定的规则，关系抽取的效果会越来越好。

5.4 本章小结

本章针对信息抽取模型模块中的命名实体识别和关系抽取模型进行了不同参数下和不同模型间的对比实验，制定了实验评估标准，并对实验结果进行分析得到了模型的最佳参数和不同模型的适用情况。

第六章总结与展望

6.1 总结

本文以上海某大数据人工智能公司的信息抽取服务系统为背景，介绍了该项目中信息抽取的两个子任务命名实体识别和关系抽取的研究现状，综述了基于深度学习的信息抽取任务所涉及的相关技术，包括：词向量、BERT模型、双向长短时记忆模型、深度残差网络、远程监督、深度学习框架Tensorflow、Docker容器技术、Kubernetes集群等。详细分析了项目中的需求并对各个模块进行了详细设计，对选择的模型结构进行分析，介绍了每个模块的具体实现过程。最后对模型的超参数进行组合对比测试，最终得到模型的参数设置。

在模型选择方面，命名实体识别模型在对比了word2vec+Bi-LSTM+CRF、BERT+Bi-LSTM+CRF、CRF、LSTM、Bi-LSTM在数据集上的表现后，本文选择了性能较好的word2vec+Bi-LSTM+CRF模型和BERT+Bi-LSTM+CRF模型。这两种模型都考虑了上下文信息的影响，也学习到了一些约束条件，有较好的分类效果。但是这两个模型的预测性能和预测时长有所不同，有各自的适用场景；关系抽取模型是在对比了CNN、CNN+Attention和ResCNN在数据集上的表现后，选择了效果更好的ResCNN。ResCNN利用了残差学习的特性使网络结构学习到了更多的语言特性从而得到了更好的效果。最后，将模型固化成pb格式再包装成app，封装进Docker容器中并将服务部署到Kubernetes集群上。业务人员可以通过访问客户端端口对信息抽取服务进行调用。

6.2 进一步工作展望

对未来的工作，我有以下几点展望：

1.对于命名实体识别模型，模型的预测性能已经可以满足基本的项目需求，希望在未来的时间里可以缩短预测时长。

2.由于项目的研发时间较短，所以关系抽取模型的效果相对来说还比较一般，希望在未来的时间里有机会在现有模型的基础上进行更多地尝试：加入Attention机制；利用PCNN代替CNN；使用新的损失函数。

3.目前的信息抽取服务还没有经历高并发场景，因为服务已经在项目里上线，希望在未来的时间里尽快对高并发的情况进行处理。

参考文献

[MUC-6, 1996] MUC-6, the Sixth in a Series of Message Understanding Conferences, was held in November 1996[OL].http://cs.nyu.edu/cs/faculty/grishman/muc6.html .

[Milleret al., 2000] Miller, Scott, Heidi Fox, Lance Ramshaw, and Ralph Weischedel. “A novel use of statistical parsing to extract information from text.” In Proceedings of NAACL, 2000.

[Bikel et al., 1999] Bikel D M,Schwarta R,Weischedel R M.An Algorithm that Learns What`s in a Name[J].Machine Learning Journal Special Issue on Natural Language Learning, 1999, 34(1-3): 211-231.

[Tsai et al., 2004] Tsai T,WU S,Lee C, et al. Mencius: A Chinese Named Entity Recognizer Using the Maximum Entropy based Hybrid Model[J]. International Journal of Computational Linguistics & Chinese Language Processing, 2004, 9(1):65-81.

[McCallum et al., 2003] McCallum A,Li W.Early Results for Named Entity Recognition with Conditional Random Fields, Features Induction and Web-enhanced Lexicons[C]. In Proceedings of the 7th Conference on Natural Language Learning at HLT-NAACL,2003: 188-191.

[张祝玉等, 2008] 张祝玉，任飞亮，朱靖波. 基于条件随机场的中文命名实体识别特征比较研究[C]. 见: 第4届全国信息检索与内容安全学术会议论文集.2008.

[Wu et al., 2015] Yonghui Wu, Min Jiang, Jianbo Lei,Hua Xu. Named Entity Recognition in Chinese Clinical Text Using Deep Neural Network. Stud Health Technol Inform. 2015;216:624-8.

[Z Huang et al., 2015] Zhiheng Huang, Wei Xu, Kai Yu. Bidirectional LSTM-CRF Models for Sequence Tagging. arXiv, 2015, 1508.01991 [cs.CL]

[Devlin J et al., 2018] 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.

[Kambhatla, 2004] Kambhatla, Nanda.”Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations.” In Proceedings of ACL, 2004.

[Zhao and Grishman, 2005] Zhao, Shubin, and RalphGrishman. Extracting relations with integrated information using kernel methods. In Proceedings of ACL, 2005.

[Culotta et al., 2006] Culotta, Aron, Andrew McCallum, and Jonathan Betz. Integrating probabilistic extraction models and datamining to discover relations and patterns in text. In Proceedings of HLT-NAACL, 2006.

[Mintz et al., 2009] Mintz, Mike, Steven Bill, RionSnow, and Dan Jurafsky. Distant supervision for relation extraction without labeled data. In Proceedings of ACL-IJCNLP, 2009.

[Socher et al. 2012] Socher , Richard, et al. Semantic compositionality through recursive matrix-vectorspaces. In Proceedings of EMNLP-CoNLL, 2012.

[Zeng et al., 2014] Daojian Zeng, Kang Liu, et al. Relation classification via Convolutional Deep Neural Network. In Proceedings of COLING, 2014

[Santos et al., 2015] Cicero Nogueira dos Santos, Bing Xiang, Bowen Zhou. Classifying Relations by Ranking with Convolutional Neural Networks. In Proceedings of ACL, 2015.

[Miwa er al., 2016] Makoto Miwa, Mohit Bansal. End-to-End Relation Extraction using LSTMs on Sequences and Tree Structures. In Proceedings of ACL, 2016.

[Lin et al., 2016] Yankai Lin, Shiqi Shen, Zhiyuan Liu, et al. Neural Relation Extraction with Selective Attention over Instances. In Proceedings of ACL, 2016.

[Bengio et al., 2002] Bengio Y, Simard P, Frasconi P. Learning long-term dependencies with gradient descent is difficult[J]. IEEE Trans Neural Netw, 2002, 5(2):157-166.

[Hinton, 1986] Hinton G E. Learning distributed representations of concepts[C]. Eighth Conference of the Cognitive Science Society. 1986.

[Harris, 1954] Harris Z S.Distributional structure[J]. Word, 1954, 10(2-3): 146-162.

[Pennington et al., 2014] Jeffrey Pennington, Richard Socher, and Christopher D. GloVe: Global Vectors for Word Representation. Manning. 2014.

[Bengio et al., 2006] Bengio Y, Ducharme R, Vincent P, et al. A neural probabilistic language model[J]. Journal of Machine Learning Research, 2006, 3(6):1137-1155.

[Mikolov et al., 2013a] Mikolov T, Chen K, Corrado G, et al. Efficient Estimation of Word Representations in Vector Space[J]. Computer Science, 2013.

[Mikolov et al., 2013b] Mikolov T, Sutskever I, Chen K, et al. Distributed representations of words and phrases and their compositionality[C]. International Conference on Neural Information Processing Systems. Curran Associates Inc. 2013:3111-3119.

[Radford et al., 2017] Radford A, Jozefowicz R, Sutskever I. Learning to Generate Reviews and Discovering Sentiment[J]. 2017.

[Rumelhart et al., 1986] David E. Rumelhart, Geoffrey E. Hinton & Ronald J. Williams. Learning representations by back-propagating errors[J]. Nature, 1986, 323(3):533-536.

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

[He K M et al., 2015] Kaiming He, Xiangyu Zhang, Shaoqing Ren, et al. Deep Residual Learning for Image Recognition[J]. arXiv, 2015, 1512.03385.

[Hinton et al., 2012] Hinton G E, Srivastava N, Krizhevsky A, et al. Improving neural networks by preventing co-adaptation of feature detectors[J]. arXiv preprint arXiv:1207.0580, 2012.

致谢

首先十分感谢我的指导老师，感谢老师在我的毕业论文编写过程中对我的指导并对我的研究方向提供了宝贵的建议，从中我学习到了很多关于论文编写和项目开发的经验与教训，使我在编写论文时更全面地描述了我的整个项目的细节。老师专业的技术知识和严谨的工作态度令我十分尊敬与佩服，是我未来学习和工作的好榜样。

感谢我在公司实习时的各位同事对我的指导与帮助，在项目开发过程中有许多问题的解决都离不开同事们的耐心指导。在项目研发中我也逐渐学到了更多分析问题和解决问题的手段与方法，我也慢慢地从学生往从业者进行过渡。

感谢学校的所有教导过我的老师们，老师们的教学态度和学术精神都是我学习的榜样，老师们传授的知识也拓宽了我的眼界，让我学习到了很多在今后的职业生涯中能够派上用场的知识，感谢老师们的辛勤付出。

感谢我的父母，我能有今天的一切都离不开你们的全力支持，希望我在求学和工作的路上可以越走越远，一帆风顺，不辜负你们的希望。

转眼之间我的研究生生涯就要结束了，今后我即将从学生转变为一个从业者，希望我在学生时代学习到的知识、完成过的项目可以学以致用，为母校争光，为自己添彩。

最后，再次感谢所有在我学习和工作中帮助过我的所有人，感谢你们的付出与帮助，也由衷地感谢在百忙之中评阅论文的各位专家和教授。

3.去除本人已发表文献复制比：去除作者本人已发表文献后，计算出来的重合字数在总字数中所占的比例

4.单篇最大文字复制比：被检测文献与所有相似文献比对后，重合字数占总字数的比例最大的那一篇文献的文字复制比

5.指标是由系统根据《学术论文不端行为的界定标准》自动生成的

6.红色文字表示文字复制部分;绿色文字表示引用部分;棕灰色文字表示作者本人已发表文献部分

7.本报告单仅对您所选择比对资源范围内检测结果负责

amlc@cnki.net

http://check.cnki.net/

http://e.weibo.com/u/3194559873/