博士学位论文

基于动态上下文相关词向量的句子级语言分析技术研究

SENTENCE-LEVEL LANGUAGE ANALYSIS WITH CONTEXTUALIZED WORD EMBEDDINGS

刘一佳

哈尔滨工业大学

2019 年 6 月

国内图书分类号：TP391.2　　学校代码：10213
国际图书分类号：681.324　　密级：公开

工学博士学位论文

基于动态上下文相关词向量的句子级语言分析技术研究

博士研究生：刘一佳

导　　师：秦兵教授

副 导 师：车万翔教授

申请学位：工学博士

学　　科：计算机应用技术

所在单位：计算机科学与技术

答辩日期：2019 年 6 月

授予学位单位：哈尔滨工业大学

Classified Index: TP391.2
U.D.C: 681.324

Dissertation for the Doctoral Degree in Engineering

SENTENCE-LEVEL LANGUAGE ANALYSIS WITH CONTEXTUALIZED WORD EMBEDDINGS

Candidate: Yijia Liu
Supervisor: Prof. Bing Qin
Associate Supervisor: Prof. Wanxiang Che
Academic Degree Applied for: Doctor of Engineering
Specialty: Computer Application Technology
Affiliation: School of Computer Science and Technology
Date of Defence: June, 2019
Degree-Conferring-Institution: Harbin Institute of Technology

摘 要

自然语言处理是人工智能的重要子学科。作为自动处理文本的第一步，将词转换为数值化表示很大程度地影响了自然语言处理的性能。词向量为自然语言的最小语义单元 — 词提供了包含句法语义信息的稠密向量表示。作为基于神经网络的自然语言处理的基础，依据词义分布假设构造的词向量给诸多自然语言处理模型带来了性能的提升。为了提高词向量的学习效率，前人工作进一步对词向量进行静态假设，即一个词有唯一的向量表示。这一假设使得在大规模数据上学习词向量成为可能，但也使静态词向量无法根据上下文环境决定其表示，因而无法建模 ‘‘一词多义’’ 等现象。动态上下文相关词向量是近年来提出的一种词向量算法。这种算法取消了静态假设并根据上下文动态地决定一个词的向量表示。在包括问答、文本蕴含、情感分析在内的多项任务中，使用上下文相关词向量的模型均取得了当前最优的性能。

很多自然语言处理任务依赖包括分词、词性标注、命名实体识别、句法分析在内的句子级语言分析。优化语言分析有助于提高自然语言处理的性能。近年来，基于神经网络的语言分析算法在静态词向量的帮助下取得了较大的性能提升。但上下文相关词向量对语言分析的作用仍有待探索。

基于上下文相关词向量与句子级语言分析技术的进展，本文围绕两者的结合开展一系列的研究，本文研究主要包括以下几方面：

1. **面向语言分析的上下文相关词向量**：针对现有上下文相关词向量使用多层网络对一整句甚至多句进行建模而导致的效率问题，本文从语言分析主要依赖局部信息的角度出发，提出一种融合相对位置权重的窗口级自注意力机制并将其应用于上下文表示，从而获得一种适用于语言分析的上下文相关词向量。五项词法句法任务的实验结果表明，由于使用局部模型替代全局模型，本文提出的上下文相关词向量在不损失精度的情况下获得了三倍的速度提升。

2. **基于上下文相关词向量的词法分析模型**：针对词法分析中的切分问题（中文分词与命名实体识别）对合理的片段（词与实体）表示的依赖，本文在上下文相关词向量的基础上提出一种基于简单拼接的片段表示方法并将其应用于半-马尔科夫条件随机场中。典型切分问题的实验结果显示本文提出的片段表示有效地提高了模型性能。通过进一步融合任务相关的上下文表示以及建模片段级信息的片段向量，本文模型取得与当前最优模型相近的性能。

3. **基于上下文相关词向量的句法分析模型**：针对上下文相关词向量对多国语句法分析作用尚无明确结论的现状，本文提出在多国语句法分析中使用上下文相关词向量并在大规模树库上验证其有效性。本文在获得显著的性能提升的基础上对提升的原因进行了详细的分析。实验结果表明，性能提升的主要原因是上下文相关词向量通过对于未登录词词形的更好的建模有效地提升了未登录词的准确率。

4. **基于知识蒸馏的句法分析加速方法**：针对使用上下文相关词向量的句法分析参数过多、运行速度较慢的问题，本文提出一种结合探索机制的知识蒸馏算法，将基于上下文相关词向量的复杂模型蒸馏到不使用相应词向量的简单模型中，从而在不显著降低性能的情况下提高句法分析速度。实验结果表明，本文提出的方法在损失少量句法分析准确率的情况下，近十倍地提升了速度。

总的来说，本文研究了动态上下文相关词向量及其在句子级语言分析中的应用。上下文相关词向量显著地提高了语言分析的性能，同时语言分析也为上下文相关词向量提供了新的理解。本文研究可以使语言分析技术更好地服务其他自然语言处理任务，从而推进整个领域的发展。

关键词：自然语言处理；上下文相关词向量；句子级语言分析技术；中文分词；依存句法分析

Abstract

Natural language processing (NLP) is an important sub-field of artificial intelligence. As the first step of automatic text processing, converting a word into its computer-understandable representation greatly affects the quality of NLP. The word vector gives the smallest semantic unit of natural language — the word a dense vector representation which contains syntactic and semantic information. As the basic block of neural-based NLP systems, the word vector which is computed according to the distributional hypothesis of words, has brought performance improvement to many natural language processing models. To improve the training efficiency, previous work to make static assumptions on word vectors, that is, a word has a unique vector. This assumption makes it possible to learn word embeddings on large-scale data, but it also makes it impossible for static word vectors to determine their representations on context, and thus cannot model the "polysemous words". The contextualized word embedding, as an emerging technique, cancels the static word vector hypothesis and dynamically determines the vector representation of a word on its context. Contextualized word embedding has helped improved a range of tasks, including questions and answers, textual implications, and sentiment analysis.

Many NLP tasks rely on sentence-level language analysis including Chinese word segmentation, part-of-speech tagging, named entity recognition, and syntactic parsing. Improving the performance of language analysis help improve the performance of natural language processing. In recent years, the help of static word embeddings help neural language analysis algorithms to achieve performance improvement. However, the role of contextualized word embeddings for language analysis is yet to be explored.

Based on the progress of contextualized word embeddings and language analysis technology, this paper focus on the combination of these two techniques. The research of this paper mainly includes the following directions:

1. **Localized and contextualized word embeddings**: Contextualized word embeddings suffer from efficiency problem, which is mainly caused by the practice of modeling the global context with multiple layers. Considering that the sequence labeling problem largely depends on the local context, we combine the relative position and the local self-attention to represent the context in the contextualized word embeddings. Experimental results on five sequence labeling problems show that, thanks to the local context

representation, our model runs three times faster over ELMo without loss of accuracy.

2. **Lexical analysis with contextualized word embeddings**: Properly representing a segment (e.g. word for Chinese word segmentation and entity for named entity recognition) is important to the segmentation performance. We propose to represent a segment by simply concatenating the input vectors and apply this representation to neural semi-CRF. Experimental results on a collection of segmentation problems show the effectiveness of our method. By incorporating additional segment embedding which encodes the entire segment, our model achieves comparable performance the state-of-the-art systems.

3. **Syntactic analysis with contextualized word embeddings**: The effect of contextualized word embeddings on universal parsing is yet to know. We propose to use the contextualized word embeddings as an additional features and show its effectiveness with experiments on extensive treebanks. We further explore the reason of performance improvement. The analysis shows the improvements are resulted from the better out-of-vocabulary word abstraction from the contextualized word embeddings and such abstraction is achieved by better morphological modeling.

4. **Distilling knowledge from syntactic parser with contextualized word embeddings**: Over-parameterization slows down both the training and testing of the model with contextualized word embeddings. We proposed a knowledge distillation method for this kind of search-based structured prediction model, which distills a complex predictor into a simple one, thus speeds up the model. Our method combines the distillation from reference states and explored state, which shows to improve the performance. Experimental results show that our method achieves magnitude of speedup with a slight loss of accuracy.

In general, this paper studies contextualized word embeddings and their applications in language analysis. contextualized word embeddings significantly improve the performance of linguistic analysis, while linguistic analysis provides a new understanding of contextualized word embeddings. This paper can make language analysis techniques better serve the downstream tasks of natural language processing, thus promoting the development of the entire field.

Keywords: Natural language processing, Contextualized embeddings, Sentence-level language analysis, Chinese word segmentation, Dependency parsing

物理量名称及符号表

为了更一致地描述上下文相关词向量及其在句子级语言分析中的应用，本文的符号系统统一如下：

- 广义符号：x，对应的词表 $\mathcal{X}$，广义符号包含字、词等；
- 广义符号的向量：$\mathbf{v}$；
- 长度为 n 的广义符号序列：$(x_1, \ldots, x_n)$，其对应的向量序列：$(\mathbf{v}_1, \ldots, \mathbf{v}_n)$；
- 包含 n 个词的句子：$(w_1, \ldots, w_n)$，对应的词向量序列采用与广义符号向量相同的方法定义 $(\mathbf{v}_1, \ldots, \mathbf{v}_n)$；
- 包含 n 个字的词：$(c_1, \ldots, c_n)$；
- 广义参数：$\mathbf{\Theta}$；
- 广义静态词向量函数：$\phi : \mathcal{X} \mapsto \mathbf{R}^d$，将一个词映射为对应的 d-维向量表示；
- 广义上下文相关词向量函数：$\mathcal{F}_c : \mathbf{R}^{n \times d} \mapsto \mathbf{R}^{n \times d'}$，将 n 个 d 维向量的序列输入；
- 线性变换函数：$\text{linear}(\mathbf{x}) = W\mathbf{x} + b$；
- softmax 函数：$\text{softmax}_{s'}(s) = \frac{1}{Z}\exp(s)$，其中 $Z = \sum_{s'} \exp(s')$；

目 录

Contents

第 1 章 绪论

1.1 课题背景及意义

1.1.1 课题背景

词向量[1–4] 是基于深度学习的自然语言处理的基础。词向量为自然语言的最小语义单元 — 词提供了包含句法语义信息的稠密向量表示。通过使用词向量作为深度神经网络的输入，一个词自身的属性（如：词性[5]），一段词之间的关系（如：命名实体[6]），两个词之间的关系（如：句法[7,8]）以及一串词组成的句子、篇章的语义主旨[9] 等都可以得到建模。词向量对于深度学习在自然语言处理中的广泛应用起到了重要的作用。

向量化一个词的语言学基础是词义的分布式假设（distributional hypothesis，文献 [10]），即：**一个词的词义可以采用它的上下文进行表示**。现今流行的词的向量化方法进一步对词向量进行了静态假设，即：**一个词有唯一的向量表示**。这种假设降低了建模的复杂度与模型的学习成本，使得在大规模数据上学习词向量成为可能。包括 Word2vec[1,2]，GloVe[3] 以及 FastText[4] 在内的一系列 ‘‘静态’’ 的词向量化算法就是这种简化的最佳实践。然而，这种假设也忽略了一个词在不同上下文环境下的句法语义差异。举例来讲，“制服’’ 在 ‘‘他制服了窃贼’’ 与 ‘‘身穿该厂制服的工人” 两种上下文环境下发挥的句法语义功能完全不同。[11] 给予不同环境中的两个词以相同的表示做法无疑是值得商榷的。

动态上下文相关词向量[12–16] 作为静态词向量的一种改进，取消了 ‘‘一个词有唯一的词向量’’ 的假设，在不同的上下文环境中，赋予相同的词不同的词向量，进而建模了一个词上下文环境相关的句法语义差异。这种词的向量化的技术在 2017 年末被提出，并在 2018 年引起广泛关注，迅速成为自然语言处理领域的热点。在其兴起的过程中，两项代表性工作扮演了重要的角色。首先，Peters 等人在 2018 年的文献 [13] 中提出基于语言模型的上下文相关词向量（Embeddings from Language Modeling，简称 ELMo）。文献 [13] 将一个词及其所在的句子输入使用长短时记忆循环神经网络（long short-term memory network，简称 LSTM，文献 [17]）建模的双向语言模型，并将得到的隐层作为对应上下文相关词向量。在文献 [13] 发表时，这种词向量帮助包括问答、文本蕴含、语义角色标注、共指消解、命名实体识别以及情感分析在内的六项自然语言处理任务中取得了当时最好的结果。在文献 [13] 发表后不久，Devlin 等人在 2018 年的文献 [16] 中提

出使用自注意力网络（self-attention network，简称 Transformer，文献 [18]）建模上下文，同时用词级别与句子级别的完形填空作为学习目标的基于自注意力网络的上下文相关词向量（Bidirectional Encoder Representations from Transformers，简称 BERT）。在 11 项任务中，使用 BERT 的模型超越了使用 ELMo 的模型，取得了当前最好的结果。种种结果均表明，上下文相关词向量是近期提出的一种提高自然语言处理性能的有效手段。

包括分词[19–21]、词性标注[5,22,23]、命名实体识别[6,24]、句法分析[8,25–28] 在内句子级语言分析问题是自然语言处理的基石。语言分析的性能很大程度上影响了包括语义角色标注[29,30]、机器翻译[31,32]、信息抽取等后续任务的性能。现阶段，基于统计学习的语言分析算法已经成为主流，而基于神经网络的方法更是应用于自然语言处理中的一种重要的统计学习手段。得益于词的成功表示，使用 ‘‘静态’’ 词向量的作为输入的语言分析模型已经在相应任务上取得了很大的成功。[5,8,21,23,28] 然而，**在语言分析中使用 ‘‘动态’’ 的上下文相关词向量尚未获得充分研究。上下文相关词向量与语言分析技术的结合中有诸多有趣并值得探索的问题。**

1.1.2 课题意义

自然语言是人机交互的一种重要手段，同时也是机器算法感知分析人类社会的一种重要途径。处理自然语言是实现机器智能的重要一步。而语言分析作为其他自然语言任务的第一步，其准确率、效率对于处理自然语言有很大的影响。同时，大部分语言分析问题衍生于语言学问题，研究语言分析问题可以帮助我们更好地研究人类语言。

时至今日，统计学习已经成为语言分析乃至整个自然语言处理的主流方法。影响统计学习模型效果的一个重要的因素是对模型输入进行合理的表示。可以说，基于统计学习的语言分析模型的发展是与表示方法的发展紧密相关的。从早期基于统计量的表示方法[33–36] 到基于大规模离散特征的表示方法[25,37–39] 再到现今流行的基于词向量的表示方法[2,5,9] 语言分析模型也从适应离散特征的线性模型[40,41] 发展到适应向量化表示的神经网路模型[17,42,43]。而作为现今广泛应用的词的表示方法，词向量的质量很大程度上影响了基于神经网络的语言分析模型的性能。[44–46] 成功建模句法语义信息的词向量能帮助语言分析模型取得更好的准确率。作为一种新兴的词表示技术，上下文相关词向量已经在一系列自然语言处理任务中表现出巨大的潜力。其对于语言分析问题的作用非常值得研究。

以语言分析问题为场景研究上下文相关词向量的意义是多方面的。其中最主要的意义是研究**通过上下文相关词向量提高语言分析模型的准确率**。为了提高准

确率，即需要研究如何将上下文相关词向量应用在语言分析问题中，也需要研究为语言分析问题设计更有效的上下文相关词向量。其次，由于大部分语言分析问题衍生于语言学问题，在语言分析问题中研究上下文相关词向量可以**分析并理解上下文相关词向量的性质**，进而指导其在其他任务中的应用。最后，将基于上下文相关词向量的语言分析模型部署到实际应用时除了要关心性能，还要关心运行效率、资源开销等方面的问题。从这些方面研究上下文相关词向量可以**使上下文相关词向量在实际场景中得到更广泛的应用**。

从上面的研究意义出发，本文拟围绕上下文相关词向量与语言分析问题开展一系列研究。1）为了通过上下文相关词向量提高语言分析的性能，本文将上下文相关词向量与现有系统融合以提高这些系统的性能，并探索基于上下文相关词向量构造自然语言结构（序列、片段、树等）表示的可能性。2）为了分析并理解上下文相关词向量的性质，本文在大规模语言分析问题上应用并分析上下文相关词向量的影响。3）为了使上下文相关词向量在实际场景中得到更广泛的应用，本文提出一种适应语言分析的高效上下文相关词向量算法。同时提出一种模型压缩的方法以实现又快又好的语言分析。

1.2 研究现状与分析

以语言分析问题为场景研究上下文相关词向量包含两方面研究主体，即：上下文相关词向量和语言分析技术。在接下来的章节，本文将回顾与分析这两个主体的研究现状。具体来讲，本文将首先回顾上下文相关词向量的前身‘‘静态/上下文无关词向量’’，将其与上下文相关词向量建立联系（§1.2.1）；然后分析现阶段主流的上下文相关词向量的算法与性能（§1.2.2）；最后结合词向量分析现阶段语言分析技术的发展情况，并讨论其与上下文相关词向量潜在的结合方向（§1.2.3）。

1.2.1 静态词向量

如前文所述，词的向量化表示是基于统计学习的自然语言处理的基础。广义上讲，所有将词转化为向量的算法都可以归类为词的向量化算法。这些算法中包括传统‘‘特征工程’’的方法，即根据一系列专家定义的启发式规则将词（以及其形态学、上下文等）特征转换为高维、稀疏、并且通常是离散的向量。随着，基于神经网络的自然语言处理的发展，越来越多的自然语言处理问题通过神经网络获得了更好的性能。所以，本文将注意力集中在基于神经网络的方法上。因而，本文关注的词的向量化的算法特指将词转化为低维、稠密、连续的向量的算法。

在讨论将词转化为低维向量表示的算法前，一个值得讨论的问题是：为什么词能够通过低维、稠密、连续的向量进行表示？从一定程度上讲，这一问题可以通

过特征工程中降维思想进行回答。降维可以将高维、稀疏、离散的具有可解释性的特征向量近似表示为低维、稠密、连续的向量。从实际应用的角度，这种降维后的向量往往具有不输于离散表示的性能，甚至在一些任务中取得了更好的性能。[47,48] 基于降维的词表示算法经验性地证明了词具有低维、稠密、连续的向量表示。基于上述论述，本文假设词具有低维、稠密、连续的向量表示，即词向量。

为了使词向量能够在实际应用中发挥作用，普遍接受的观点是词向量应该携带词的句法语义信息。[2,44,45] 这种句法语义信息通常是通过建模词义的分布式假设来实现的。接下来，本文将从建模词义分布式假设的角度出发，分析一系列**静态/上下文无关**的词向量算法。①

1.2.1.1 使用 ‘‘共现矩阵’’ 建模分布式假设的方法

词义分布式假设强调 ‘‘上下文相似的词有相似的词义’’。词义分布式假设包含两部分主体：一个是上下文，另一个是被描述的词（下文称“目标词’’）。建模词义分布式假设本质上是：1）选择一种方式描述上下文；2）选择一种模型描述目标词与其上下文之间的关系。

最简单的描述目标词与上下文关系的方法是关注目标词与哪些词或文档共现。潜在语义分析（Latent semantic analysis，简称 LSA）是这类方法的代表。最早，LSA 提出的目标是建模信息检索中词义相似的词。LSA 中的一个目标词首先被表示为一个向量。向量的维度与文档的个数相等。一个词的向量的某一维表示这个词在对应文档中的某种计量。② 然后，所有词被堆叠成为一个矩阵，即共现矩阵。最后经过矩阵分解技术，原始的词向量被映射到低维的表达词的潜在语义的空间。奇异值分解（Singular Value Decomposition，简称 SVD）是这类方法中最常用的矩阵分解技术。从建模分布式假设的角度来看，LSA 使用目标词所在的文档描述目标词的上下文。

除了使用文档间接地描述上下文，目标词周围的词也可以描述上下文，从而构造共现矩阵。Lund 与 Curt 在 1996 年的文献 [49] 中提出 Hyperspace Analogue to Language（简称 HAL）算法，首次使用目标词与其周围的词的共现次数建模目标词的词义（如图1-1所示）。Rohde 等人在 2006 年的文献 [50] 中提出 COALS 算法并对 HAL 进行了一系列改进。这些改进包括：将共现矩阵中的计数转化为皮尔逊相关系数（Pearson’s correlation）并去掉负相关的共现，同时使用 SVD 分解获得词的低维的向量化表示。文献 [50] 可以认为是使用目标词与周围词的关系计算低维、

① 这里的 ‘‘上下文无关’’ 是从一个词向量化的过程是否使用其上下文的角度出发的。由于词向量的学习过程必定依赖其上下文，根据词向量的学习过程区分是否上下文相关并无太大意义。

② 这种计量可以是频次，tf-idf，点互信息等。

	barn	fell	horse	past	raced	the
barn	1	1	0	0	0	1
fell	1	1	0	0	0	0
horse	0	0	1	0	1	1
past	0	0	0	1	1	1
raced	0	0	1	1	1	0
the	1	1	0	1	0	2

图 1-1 使用词的上下文窗口构造共现矩阵的一个例子[49]。这个例子为 “the horse raced past the barn fell’’ 建立了窗口大小为 2 的共现矩阵。

Fig. 1-1 An example of the concurrence matrix which models the sentence “the horse raced past the barn fell” with a window of 2 words.

连续、稠密词向量的先驱工作。在这之后，多种基于共现矩阵建模词义分布式假设的词向量算法被提出。其中包括 Hellinger PCA 算法 [51]，GloVe 算法 [3] 等。值得一提的是，Pennington 等人在 2014 年的文献 [3] 中提出的 GloVe 算法也使用共现矩阵建模了词义分布式假设。但不同于采用矩阵分解降维的工作，GloVe 算法首先假设目标词存在低维向量化表示，然后将向量化的表示输入一个函数并用回归的方式对共现矩阵中的值进行拟合以达到降维（或向量化）的效果。

Word2vec[2] 是另一个流行的向量化算法。Word2vec 算法可以概括为将目标词的向量化的表示输入一个神经网络预测其在生语料中的上下文。可以说，Word2vec 是使用 ‘‘预测目标词或上下文’’ 的思路建模分布式假设。但 Levy 等人在 2014 年的文献 [52] 中证明使用负采样（negative sampling）优化的 Word2vec 等价于对于目标词及其窗口内上下文的点互信息矩阵进行 SVD 分解。这一发现将基于 ‘‘共现矩阵’’ 与基于 ‘‘预测目标词或上下文’’ 的方法进行了统一。后文会从 ‘‘预测目标词或上下文’’ 的角度对 Word2vec 进行讨论。

1.2.1.2 使用 ‘‘预测目标词’’ 建模分布式假设的方法

除了通过目标词 w_i 与其上下文 $\{w_j\}$ 的共现描述两者关系，通过上下文预测目标词或者 通过目标词预测上下文也是一种描述两者关系的常用方法。这种方法的第一步是定义一个词到其向量的映射关系 ϕ。将一个词 w_i 及其上下文的 w_j 输入到 ϕ 中获得他们的向量表示 $\mathbf{v}_i$ 与 $\{\mathbf{v}_j\}$，并把这些向量输入到一个建模上下文的函数 $\mathcal{F}_c$；从而获得其上下文表示 $\mathbf{h}_i = \mathcal{F}_c(\{\mathbf{v}_j\})$，最后将 $\mathbf{h}_i$ 输入到一个预测函数 $\mathcal{F}_p(\mathbf{h}_i, \mathbf{v}_i)$ 达到描述两者关系的目标。在这类方法中，ϕ 是词向量算法的最终产出。

基于语言模型的方法 前人工作探索了不同的上下文函数 $\mathcal{F}_c$ 与不同预测函数 $\mathcal{F}_p$。一种典型的组合是使用前文表示上下文，根据前文预测下一个词，即语言模型。Bengio 等人在 2003 年的文献 [54] 中提出一种利用有限窗口上文预测下一个词的

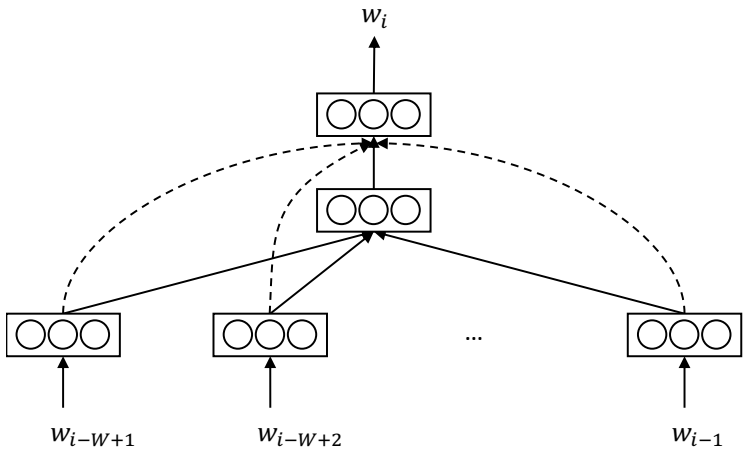

图 1-2 Bengio 等人在 2003 年的文献 [53] 中提出的语言模型

Fig. 1-2 The language model proposed by Bengio et al. (2003) [53]

语言模型（如图1-2所示），成为这一类方法的先驱。具体来讲，给定一个窗口大小为 W 的上文 $w_{i-W-1},\ldots,w_{i-1}$ 以及目标词 w_i，文献 [54] 使用向量拼接的方式建模 $\mathcal{F}_c$，并使用多层神经网络建模用上文预测下一个词的概率 $\mathcal{F}_p = p(w_i \mid w_{i-W-1},\ldots,w_{i-1})$。其模型可以形式化地描述为，

$$\mathbf{h}_i = \oplus_{j=i-W-1}^{i-1}\mathbf{v}_j$$

$$\mathbf{s} = \text{linear}^{'}(\mathbf{h}_i) + \text{linear}(\tanh(\text{linear}(\mathbf{h}_i)))$$

$$p(w_i \mid w_{i-W-1},\ldots,w_{i-1}) = \underset{w}{\text{softmax}}\left(\mathbf{s}_{w_i}\right)。$$

其中，$\oplus$ 代表向量的顺序拼接。对 Bengio 的模型中的预测函数 $\mathcal{F}_p$ 与上下文函数 $\mathcal{F}_c$ 进行了一系列改进。这些改进包括：Mnih 与 Hinton 在 2007 年的文献 [55] 中提出的在建模上下文 $\mathcal{F}_c$ 时使用加权求和取代拼接的 Log-bilinear Language Model（简称 LBL），以及 Mnih 与 Hinton 在 2007 年的文献 [56] 中提出的使用基于词表聚类的启发式负采样提高 LBL 中预测函数 $\mathcal{F}_p$ 的学习速度的 Hierarchical Log-Bilinear Language Model。

除了使用简单拼接或者加权平均的方法建模上下文 $\mathcal{F}_c$，循环神经网络（recurrent neural network，简称 RNN）具备建模任意长度历史的能力，因而也可以作为上下文的一种建模。Mikolov 等人在 2010 年的文献 [57] 中提出 RNN 语言模型建模词的上下文（如图1-3所示）。与文献 [54] 的思路一样，RNN 语言模型也尝试建模一定上文条件下一个词出现的概率。具体来讲，其形式如下：

$$\mathbf{v}_i = \text{RNN}(\mathbf{v}_1,\ldots,\mathbf{v}_{i-1})$$

$$\mathbf{s} = \text{linear}(\mathbf{v}_i)$$

$$p(w_i \mid w_1,\ldots,w_{i-1}) = \underset{w}{\text{softmax}}\left(\mathbf{s}_{w_i}\right)。$$

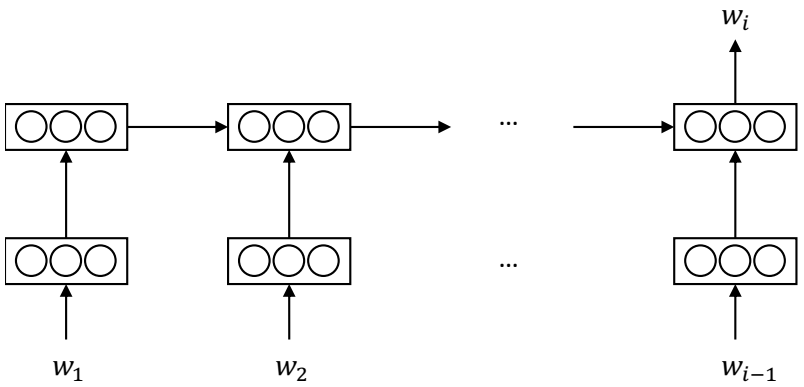

图 1-3 基于循环神经网络的语言模型

Fig. 1-3 The recurrent neural network language model

可以说，语言模型是最早的成功建模上下文以及上下文与目标词关系的方法。通过分析可见，语言模型的思想在在上下文相关词向量中也得到了应用。

基于完形填空的方法 Collobert 等人在 2011 年的文献 [43] 中提出了另一种基于‘‘完形填空’’的描述上下文以及上下文与目标词关系的方法，即包含目标词的上下文的‘‘真实程度’’大于将目标词替换为随机词的上下文的真实程度。具体来讲，对于窗口为 $2W$ 的上下文，文献 [43] 使用卷积神经网（convolutional neural network，简称 CNN，文献 [42]）建模上下文 $\mathcal{F}_c$，并将上下文向量输入一个前馈神经网络中获得一个标量 s 作为这一上下文‘‘真实程度’’的一种度量。具体来讲，一段上下文 $\mathbf{v}_{i-W},\ldots,\mathbf{v}_i,\ldots,\mathbf{v}_{i+W}$ 的‘‘真实程度”可以描述为：

$$\begin{aligned}\mathbf{h}_i &= \mathrm{CNN}(\mathbf{v}_{i-W},\ldots,\mathbf{v}_i,\ldots,\mathbf{v}_{i+W})\\ s_{w_i} &= \mathrm{linear}'(\tanh(\mathrm{linear}(\mathbf{h}_i)))\text{。}\end{aligned} \tag{1-1}$$

为了训练模型参数，文献 [43] 使用最大化分类边界学习目标（margin loss）：

$$\boldsymbol{\Theta} = \underset{\boldsymbol{\Theta}}{\operatorname{argmin}} \max\{0, 1 - s_{w_i} + s_{w'}\} \tag{1-2}$$

建模了这种完形填空的训练思想。其中 $s_{w'}$ 为将在上下文 $w_{i-W},\ldots,w_i,\ldots w_{i+W}$ 中的 w_i 替换为 w' 后重新计算公式1-1获得的分数。

文献 [43] 提出的完形填空的思想有诸多独到之处。相对语言模型来讲，其最为重要的不同是在描述上下文时不只考虑上文，更可以考虑下文。可以说，这种思想与 Devlin 等人在文献 [16] 中提出的 BERT 上下文相关词向量有紧密的联系。

基于单个上下文词与目标词关系的方法 如前文所述，建模词义分布式假设的核心思想由两部分组成：1）选择一种方式描述上下文；2）选择一种模型刻画目标词

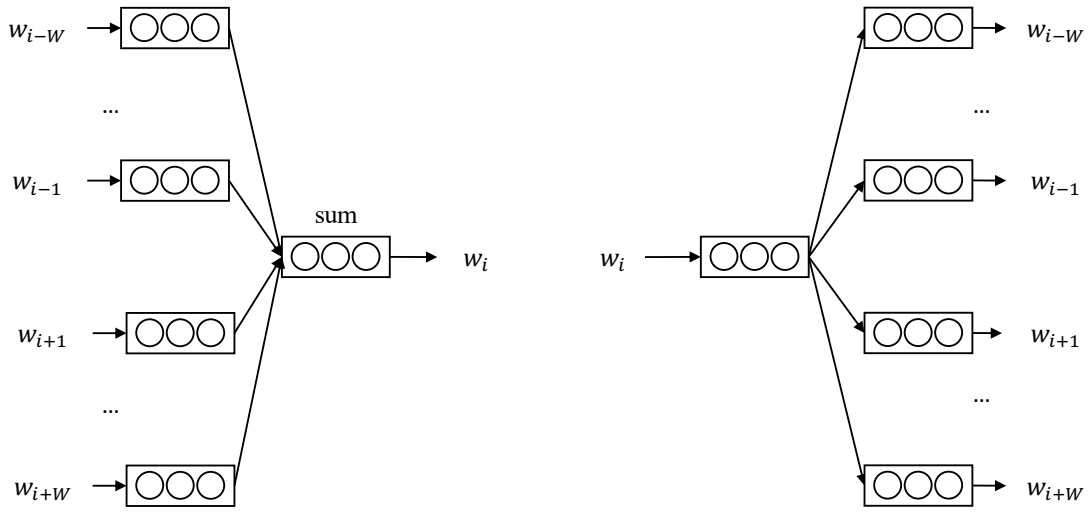

图 1-4 文献 [1] 提出的两种模型：CBOW（左）与 Skip-gram（右）

Fig. 1-4 The CBOW model (left) and Skip-gram model (right) from Mikolov et al. (2013) [1]

与其上下文之间的关系。使用语言模型以及使用完形填空的方法都对一定宽度内上下文整体进行了表示。能否将上下文的整体表示化简为单个独立的词并预测单个上下文词与目标词之间的关系呢？Mikolov 等人在 2013 年文献 [2] 中基于这种思想提出了 Word2vec 模型。

相较语言模型，Word2vec 模型采用某个上下文词的词向量 $\mathbf{v}'_j$ 作为 $\mathcal{F}_c$，并使用目标词预测这个上下文或这个上下文预测目标词的概率作为 $\mathcal{F}_p$。文献 [2] 提出两种模型建模，分别是 CBOW 与 Skip-gram 模型（参考图1-4）。CBOW 建模的是根据上下文词的评价预测目标词的概率，即 $p(w_i \mid w_{i-W},\ldots,w_{i-1},w_{i+1},\ldots,w_{i+W})$；Skip-gram 建模的是根据目标词预测某个上下文的概率，即 $p(w_j \mid w_i)$，其中 $i-W \le j \le i+W, j \ne i$。具体来讲，CBOW 模型可以描述为

$$\mathbf{h}_i = \frac{1}{2n} \sum_{i-W \le j \le i+W, j \ne i} \mathbf{v}'_j$$

$$p(w_i \mid w_{i-W},\ldots,w_{i-1},w_{i+1},\ldots,w_{i+W}) = \underset{w}{\text{softmax}}\left((\mathbf{v}_i)^T \cdot \mathbf{h}_i\right)。$$

而 Skip-gram 可以描述为

$$p(w_j \mid w_i) = \underset{w}{\text{softmax}}\left((\mathbf{v}_i)^T \cdot \mathbf{v}'_j\right)。 \tag{1-3}$$

需要说明的是，文献 [2] 采用两套不同的词向量分别建模了上下文词向量与预测词向量。

Word2vec 算法作为基于神经网络的自然语言处理模型的基础对于自然语言处理的影响是深远的。但 Word2vec 对于词、上下文以及外部知识的过度简化也激发了一系列工作。1）由于 Word2vec 在建模词时没有考虑词的形态学特征，一系列工

作尝试将词的形态学信息融入 Word2vec 中。FastText[4] 是这类方法中最典型的代表。相比 Word2vec 采用的给每个词一个词向量的做法，FastText 给词内字级别的 ngram 一个向量，并用 ngram 向量相加的方式获得词向量。相较 Word2vec，FastText 可以给任意词一个向量，因而具有处理未登录词的能力。2）由于 Word2vec 没有考虑目标词与上下文词的相对位置，Wang 等人在文献 [58] 中提出使用不同的 ϕ 表示不同相对位置的词，从而改进了原始算法。3）由于 Word2vec 在建模词义时没有考虑人工构建的词义词典，一系列工作也尝试将外部知识，比如 WordNet[59]，双语词典[60]，平行语料[61]，远程监督信息[62] 等，引入 Word2vec 的模型学习中。

静态词向量的评价 至此，本章已经对当前具有代表性的 ‘‘静态’’ 词的向量化算法进行了回顾。一个自然产生的问题是比较这些向量化的算法的优劣。Lai 等人在 2016 年的文献 [45] 中对这些 ‘‘静态’’ 词向量化算法从多个方面进行了比较。文献 [45] 比较的内容包括不同的因素对于产生的词向量的质量的影响。文献 [45] 采用语义相似度、语义类比，用作特征时的模型性能，用作初始化时的模型性能三类指标评价了不同词向量的性能。这一系列工作尝试回答：1）训练语料规模对于模型性能的影响；2）相同的参数设置下，哪种模型性能更好。对于第一个问题，文献 [45] 的结论是是训练数据规模越大，得到的词向量性能越好。对于第二个问题，他们的研究没有给出明确的答案。不同的词向量算法在不同的实验指标下表现并不一致。Yin 与 Schutze 也在 2016 年的文献 [63] 中验证了这一观察。

值得一提的是，词向量化的评价是一个争议很大的问题。词义相似度是很多词向量算法使用的评价指标。然而，Faruqui 等人在文献 [44] 中指出词义相似度有诸多问题，比如：词义相似度的评价具有主观性，与特定任务关联较强，与下游任务的性能相关性较弱，不能评价一词多义的现象等。

本文关注的主要问题是词向量与下游语言分析任务的关系，一般来讲，词向量在神经网络中的应用一般有两种模式：1）*做为特征*，以及 2）*做为神经网络的初始化*。第一种模式将词向量用作输入，并在模型学习中保持不变。第二种模式用词向量初始化参数。第一种模式使模型可以给未出现在训练数据中的词（指的是特定任务的训练数据）一个合理的词向量。这种方式往往给模型带来更好的泛化性，但由于词向量的泛化能力问题，也给模型学习带来了困难。第二种模式往往使模型更容易学习，但也牺牲了一定的泛化性。两种方式孰优孰劣无法从理论上给出解释，大部分工作是依靠实际任务的性能进行评价的。

关于不同的词向量在实际任务中的表现一直众说纷纭。文献 [45] 与文献 [63] 都讨论了这一问题。表1-1总结了他们的比较结果。两篇文献都在相同的数据规模

表 1-1 前人工作中的不同词向量的性能
Table 1-1 A comparison on the static word embedding algorithms

	上下文表示	目标词与上下文的关系	文献 [45] NER	文献 [45] POS	文献 [63] POS
Random			84.39	95.41	-
GloVe	加权的共现比例	预测共现比例	88.19	96.42	96.65
Skip-gram	上下文某个词的词向量	预测目标词	**88.90**	96.57	-
CBOW	上下文各词词向量的平均值	预测目标词	88.47	96.63	96.40
LBL	上下文各词的语义组合	预测目标词	88.69	**96.77**	-
HLBL	上下文各词的语义组合	预测目标词	-	-	96.53
NNLM	上下文各词的语义组合	预测目标词	88.36	96.73	-
C&W	上下文各词与目标词的语义组合	目标词与上下文的联合打分	88.15	96.66	96.58

的设置下重新训练了词向量。其中，文献 [45] 在 CoNLL03 命名实体识别数据集[64] 上将词向量用作特征，在 Penn Treebank （简称 PTB）词性标注数据集[65] 上将词向量用作初始化。文献 [63] 在 PTB 词性标注数据集上将词向量用作初始化。从表1-1可以看出，不同的词向量的性能存在一定差异，但这种差异并不显著，同时某种词向量并不稳定地好于其他词向量。究其原因，本文认为主要的问题在于这类方法采用静态的词向量表示方式，即一个词只有唯一的向量化表示。这种表示方式使得词向量无法显示地建模一个词在不同上下文环境下句法语义功能的不同。

1.2.2 上下文相关词向量

一个词在不同的上下文环境下发挥的句法语义功能可能存在很大差异。举例来讲，‘‘制服’’ 在 ‘‘他制服了窃贼’’ 与 ‘‘身穿该厂制服的工人” 两种上下文环境下分别用作动词（代表 ‘‘用强力使之驯服’’）以及名词（代表 ‘‘有规定式样的服装’’）。给两种上下文环境下的 ‘‘制服’’ 相同的向量表示的做法无疑是值得商榷的。为了建模这种与上下文相关的句法语义功能，更合理的向量化方法是根据一个词及其上下文环境动态地决定它的向量表示。接下来，本章将按照上下文相关词向量的发展过程对前人研究进行回顾。

context2vec Melamud 等人在 2016 年的文献 [66] 中首次从词向量的角度讨论这一问题并提出 context2vec 算法。context2vec 在描述预测目标词与上下文关系时可以类比 CBOW 模型。但不同的是，context2vec 采用 LSTM 描述上下文 $\mathcal{F}_c$。具体来讲，context2vec 将从左向右以及从右向左的词分别输入两个 LSTM 中，并将隐层输出作为上下文的表示。context2vec 可以形式化地定义为：

$$\mathbf{h}_i^* = \overrightarrow{\text{LSTM}}(\mathbf{v}_1, \ldots, \mathbf{v}_{i-1}) \oplus \overleftarrow{\text{LSTM}}(\mathbf{v}_{i+1}, \ldots, \mathbf{v}_n)$$

$$\mathbf{h}_i = \text{linear}' \left(\text{ReLU}\left(\text{linear}\left(\mathbf{h}_i^*\right)\right)\right)$$

$$p(w_i \mid w_1, \ldots, w_{i-1}, w_{i+1}, \ldots, w_n) = \underset{w}{\text{softmax}}\left((\mathbf{v}_i)^T \cdot \mathbf{h}_i\right)。$$

除了选择不同的方式描述上下文，相对 Word2vec 一类的静态词向量，context2vec 最重要的不同在于**采用 $\mathbf{h}_i$ 作为 w_i 的词向量**。这种做法使得同一个词在不同的上下文条件下拥有不同的向量表示。可以说，context2vec 是上下文相关词向量的最早尝试。

CoVe McCann 等人在 2017 年的文献 [12] 中再次强调了依照上下文对词进行向量化的重要性。在文献 [12] 中，McCann 等人使用神经网络的机器翻译[67–69] 的解码器的输出作为对应词的上下文相关词向量，并将其算法命名为上下文相关词向量（Contextualized word vectors，简称 CoVe）。虽然并未明确定义目标词与上下文的，可以认为 CoVe 使用双向 LSTM 描述上下文 $\mathcal{F}_c$，用机器翻译中目标语的词作为目标词，从而使用翻译的关系描述了源语言上下文与目标语词之间的关系。McCann 等人将 CoVe 词向量作为特征输入，并在情感分析[70]、问题分类[71]、文本蕴含[72] 以及阅读理解[73] 中取得了性能的提升。

ELMo 从建模词义分布式假设的角度讲，CoVe 并未对目标词以及上下的关系进行明确定义。2018 年，Peters 等人在文献 [13] 中明确将目标词定义为语言模型中待预测的下一个词，并提出基于语言模型的上下文相关词向量— ELMo。具体来讲，一个句子 $(w_1, \ldots, w_n)$ 的前向语言模型可以形式化地定义为：

$$p(w_i \mid w_1, \ldots, w_{i-1}) = \underset{w}{\text{softmax}}\left(W_{w_i} \overrightarrow{\mathbf{h}}_{i-1} + b_{w_i}\right)。 \tag{1-4}$$

其中 $\overrightarrow{\mathbf{h}}_i$ 是前向上下文表示模型的最终输出。后向语言模型采用相同的方式定义。

为了建模语言模型并计算 $\overrightarrow{\mathbf{h}}_i$，文献 [13] 首先使用基于字的 CNN 计算一个词的上下文无关的表示，即 $\mathbf{v}_i = \text{CNN}(w_i)$；然后使用带有短路机制（skip-connections，文献 [74]）的多层 LSTM 表示前向/后向上下文。以前向语言模型为例，多层模型可以递归地定义为

$$\overrightarrow{\mathbf{h}}_i^{(k)} = \overrightarrow{\text{LSTM}}^{(k)}\left(\overrightarrow{\mathbf{h}}_1^{(k-1)}, \ldots, \overrightarrow{\mathbf{h}}_i^{(k-1)}\right)。 \tag{1-5}$$

其中，$\overrightarrow{\mathbf{h}}_i^{(0)} = \mathbf{v}_i$ and $\overleftarrow{\mathbf{h}}_i^{(0)} = \mathbf{v}_i$；最后，上下文相关词向量可以定义为 L 层隐层状态的加权平均，即

$$\text{ELMo}_i = \gamma \sum_{k=0}^{L} s_k \cdot \left(\overrightarrow{\mathbf{h}}_i^{(k)} \oplus \overleftarrow{\mathbf{h}}_i^{(k)} \right) 。 \tag{1-6}$$

文献 [13] 提出通过最大化下一个词的对数似然的方式在大规模生文本中学习语言模型。最终学习目标是两个方向上的对数似然的加和，形式化地定义为：

$$\begin{aligned} \boldsymbol{\Theta} &= \underset{\boldsymbol{\Theta}}{\text{argmax}} \sum_{i=1}^{n} (\log p(w_i \mid w_1, \ldots, w_{i-1}) + \log p(w_i \mid w_{i+1}, \ldots, w_n)) \\ &= \underset{\boldsymbol{\Theta}}{\text{argmax}} \sum_{i=1}^{n} \log \underset{w}{\text{softmax}} \left(\text{linear} \left(\overrightarrow{\mathbf{h}}_i \right) \right) + \log \underset{w}{\text{softmax}} \left(\text{linear} \left(\overleftarrow{\mathbf{h}}_i \right) \right) 。 \end{aligned} \tag{1-7}$$

文献 [13] 将上下文相关词向量用作特征，在包括阅读理解[73]、文本蕴含[72]、语义角色标注[75]、共指消解[76]、命名实体识别[64]、以及情感分析[70] 在内的六项数据集上取得了当时最优结果。

除了性能的提升，文献 [13] 也指出对于多层 LSTM，使用语言模型为目标学习出的网络具有从句法到语义的逐步抽象能力。具体来讲，LSTM 的底层有更好的句法抽象能力，高层具有更好的语义抽象能力。① 同年，Peters 等人在文献 [77] 中对于 ELMo 进行了更加深入的分析。根据文献 [77]，ELMo 的成功主要来自从大规模生语料中使用语言模型的学习目标学习参数。表示上下文的网络结构对于最终模型性能的影响并不大。② 同时，不同的网络都表现出底层的句法抽象能力更好，高层的语义抽象能力更好的特性。

ULMFit 在 ELMo 同时，Howard 与 Ruder 在 2018 年的文献 [14] 中从迁移学习的角度研究了从语言模型迁移到文本分类任务的最佳实践方法。文献 [14] 强调前人有关语言模型未能给文本分类带来显著性能提升的主要原因是并未合理地对语言模型进行精调参数（fine-tune parameters）。文献 [14] 从另一个角度佐证了上下文相关词向量的有效性。

GPT 可以看出，从描述上下文到描述目标词与上下文的关系，ELMo 都与第1.2.1.2节中基于 RNN 的语言模型有密切关系。不同于前人工作将 RNN 语言模型中的词向量表视作向量化的最终产物，ELMo 将整个 RNN 视作是向量化的产物并取得了巨大的成功。除了观念的变化，近年来语言模型技术的发展[53,57,79,80] 以及多任务学习技术的发展[81,82] 也为 ELMo 的成功提供了保障。沿着基于语言模型的上下文相关词向量的思路，Radford 等人在 2018 年的文献 [15] 中

① 实践中，底层在词性标注等典型句法任务中表现更好，高层在共指消解等典型语义任务中表现更好。

② 文献 [77] 尝试了 Gated CNN[78]，Transformer[18] 两种网络结构。

提出使用单向 Transformer 建模上文的模型 — Generative Pre-Training（简称 GPT）。除了描述上下方式的不同，GPT 的最大特点是在特定任务上对预训练的语言模型进行精调参数。这种方法进一步提升了模型性能。

BERT 2018 年下半年，Devlin 等人基于 GPT 预训练的思想，在文献 [16] 中提出使用自注意网络（Bidirectional Encoder Representations from Transformers，简称 BERT）建模目标词的左侧和右侧的上下文。通过将目标词遮罩并使用词级别完形填空（或称遮罩语言模型，masked language model）和句子级完形填空的学习目标训练双向自注意网络，从而取得了 11 项任务的大幅度性能提升，获得了超越 ELMo 与 GPT 的效果。

相对 ELMo 与 GPT，BERT 对于建模词义分布式假设做出的改进是多方面的。从描述上下文的角度，BERT 利用双向自注意网络的特点，真正地描述了 ‘‘双向’’ 的上下文。从描述目标词与上下的角度，BERT 提出的两个学习目标如下：

• 词级别完形填空：随机将句子中的词替换为一个特殊标记，并用这个位置的上下文相关词向量预测原来的词；

• 句子级完形填空：将两个句子连接起来，并预测第二个句子是不是第一个句子的后续句子。值得一提的是，这种句子级完形填空的分类问题是在词级别上进行预测与学习的。

这种方式与语言模型有很大差异。但与 Collobert 等人 2011 年的工作 [43] 有相似之处。

通过上述讨论不难看出，虽然产生巨大变革，上下文相关词向量仍可以套用建模词义分布式假设的两个维度：上下文以及上下文和目标词的关系进行解释。表1-2从这两个维度对上述上下文相关词向量方法进行总结。然而，上下文相关词向量的研究不应只局限于表1-2。新的上下文表示方法以及新的建模目标词与上下文关系的模型都是值得探索的。

1.2.3 基于词向量的语言分析技术

包括分词、词性标注、命名实体识别、句法分析在内的语言分析任务是下游自然语言处理任务的基础（如图1-5所示）。语言分析模型的性能很大程度上影响了下游任务的准确率。因而吸引了大量科研兴趣。在接下来的章节中，本章将首先从词法分析（§1.2.3.1）和句法分析（§1.2.3.2）两个方面对于当前研究进行回顾。然后讨论表示学习与基于结构预测的语言分析的关系（§1.2.3.3）。

1.2.3.1 基于词向量的词法分析

一般认为，词法分析包括中文分词、词性标注以及命名实体识别。其中，中文

表 1-2 上下文相关词向量算法的比较。

Table 1-2 A comparison on the contextualized word embedding algorithm.

模型名称	上下文表示	目标词与上下文的关系	用法	Multi-NLI 结果
CoVe	双向 LSTM	机器翻译	特征	-
ELMo	双向多层 LSTM	双向语言模型	特征	79.6/79.3
ELMo+	双向多层 Gated CNN	双向语言模型	特征	78.3/77.9
	双向 Transformer	双向语言模型	特征	79.4/78.3
GPT	Transformer	单向语言模型	初始化	81.8/-
BERT	Transformer	词级别完形填空，句子级完形填空	初始化	86.7/85.9

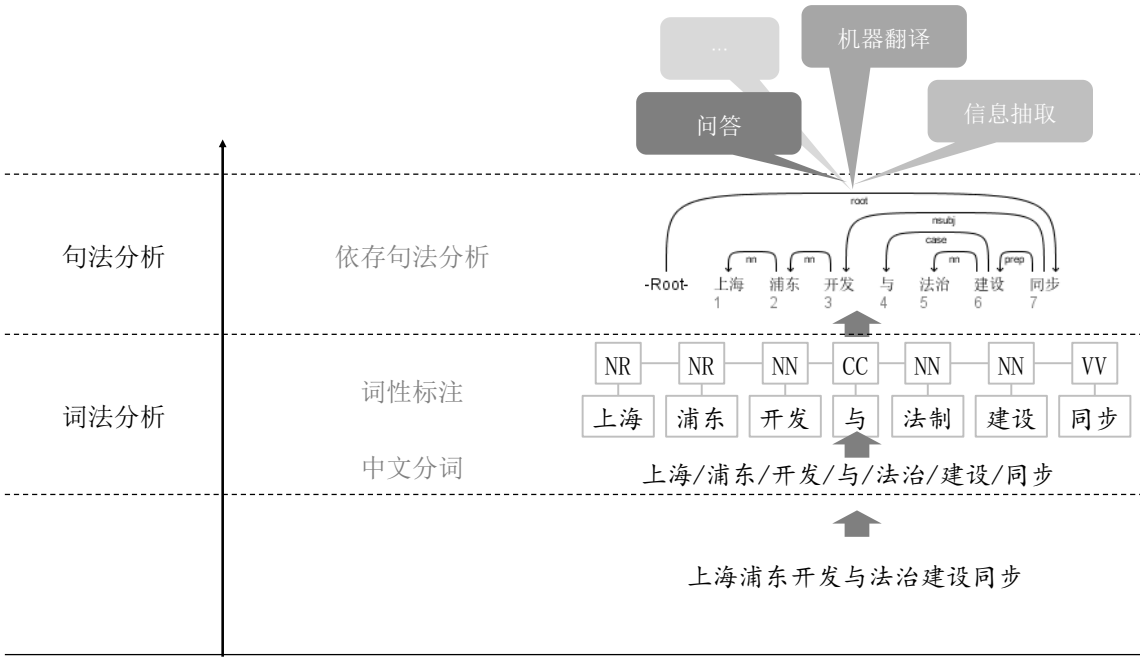

图 1-5 一个语言分析的例子

Fig. 1-5 An example of language analysis

分词指的是把中文的原始句子分割成词序列的过程。举例来讲，对于句子‘‘浦东开发与经济建设同步’’进行分词可以得到“浦东 开发 与 经济 建设 同步’’的词序列。词性标注指的是为句子中的每个词标注词性的过程。词性主要包括名词、动词、形容词、副词、介词、代词等，是一个词的句法类别的概括。命名实体识别指的是识别句子中的实体并给实体打标签的过程，以“Michael Jordan is a professor at Berkeley’’为例，命名实体识别将连续的词片段“Michael Jordan’’识别为人名，将“Berkeley’’识别为机构名。根据输出结构的不同，这三种问题可以归类为切分（分词和命名实体识别）以及序列（词性标注）两类问题。

对于切分问题，最直接的建模方法是将其转换为标注边界的序列问题[19,24]。以“Michael Jordan is a professor at Berkeley’’的命名实体识别为例，这种方法给输入

的 7 个词打 “B-PER, E-PER, O, O, O, O, S-ORG’’ 标签。这种建模范式将三种词法分析进行了统一。从这种统一的序列标注的角度来看，词法分析模型可以拆解为两部分：1）对输入的词进行表示；2）对输出的序列结构进行建模。Collobert 等人在 2011 年的文献 [43] 中最早将词向量输入卷积神经网络从而对于词及其周围的上下文进行表示。文献 [43] 使用简单分类的方式对结构进行建模。Huang 等人在 2015 年的文献 [5] 将基于 LSTM 的上下文表示引入词法分析，并使用 CRF 建模结构。Lample 等人在 2016 年的文献 [6] 中强调了使用基于字的模型对词进行表示，从而取得显著的性能提升。Peters 等人在 2017 年的文献 [83] 中提出将语言模型的输出作为词表示的补充，从而获得进一步的性能提升。

从基于词向量的一系列词法分析工作来看，模型性能的提升总是伴随着词表示方法的进步。Reimers 与 Gurevych 也在 2017 年文献 [84] 中指出词法分析模型的性能与词向量选取的相关性。但文献 [84] 显示合理建模结构仍有其意义。

除了使用 CRF 对结构建模，前人工作也尝试了在词法分析中使用基于转移以及全局解码的方法[85,86]。可以说，加强词表示是词法分析性能提升的重要途径。在词表示相同的条件下，合理地建模结构也能带来性能的提升。

1.2.3.2 基于词向量的句法分析

主流句法表示形式有两种：短语句法和依存句法。短语结构句法描述短语的句法功能以及若干短语组合形成新的短语的过程。依存句法则描述句子中词与词的修饰关系。相较短语结构句法，依存句法具有形式简单易懂、可以刻画交叉关系等优势。因而，本文将句法分析的重点放在依存句法上。

依存句法分析的算法主要有两类：基于图的方法与基于转移的方法。基于图的方法将依存句法分析转化为一个寻找最大生成树问题。如果打分函数定义在有限阶的弧上，寻找最大生成树的问题可以通过动态规划进行求解[25]。基于转移的方法则将依存句法分析转化为一个最优移进-规约序列的搜索问题[26]。

Chen 与 Manning 在 2014 年的文献 [28] 中首次将词向量应用于基于转移的依存句法中。文献 [28] 的方法在提出之处尚无法与传统基于特征的方法抗衡。但 Weiss 等人在 2015 年的文献 [87] 发展了文献 [28] 的方法，并通过多层网络强化了词表示并取得了当时最优的性能。然而，文献 [87] 的方法非常依赖调参且稳定性较差。Dyer 等人在 2015 年的文献 [7] 中提出的通过 LSTM 建模词的上下文的方法真正给基于词向量的句法分析带来了稳定的提升，进而证明有效表示词对于句法分析的重要性。

上述工作主要针对基于转移的方法。Kiperwasser 与 Goldberg 在 2016 年的文献 [88] 中首次将词向量应用于基于图的依存句法分析中。文献 [88] 使用双向 LSTM

对上下文合理地建模并在此基础上使用一阶 Esiner 算法寻找最大生成树。文献 [88] 获得了相对基于转移的方法更好的性能。而 Dozat 与 Manning 在 2017 年的文献 [8] 中进一步发展了文献文献 [88] 的方法。相较文献 [88]，文献 [8] 强化了上下文表示并弱化结构预测算法。其采用一种贪心的近似算法计算最大生成树。这种简化为模型性能带来了显著提升。之后，Dozat 等人在 2017 年的文献 [89] 中加入基于字的词表示，进一步提升了性能。

通过回顾前人基于词向量的句法分析方法，本文将其性能提升的关键总结为对于词表示的强化。这一观察与前一节相符，即词表示能力的提升是句法分析性能提升的关键。

1.2.3.3 表示学习与基于结构预测的语言分析的关系

从结构预测的观点看待这些语言分析问题，每种问题都有特殊的结构与之对应。其中，分词、命名实体识别对应分割结构预测问题；词性标注对应序列结构预测问题；句法分析对应树结构预测问题。在神经网络兴盛之前，研究人员热衷于对自然语言结构进行复杂的建模。这类研究趋势可以进行如下总结：1）*强调全局解码*：从最大熵马尔科夫模型[90] 到条件随机场模型[91] 是这一研究范式的代表。这类研究强调在全局条件下无偏地求最优结构（应用于测试阶段）以及无偏地计算边缘概率（应用于训练阶段）。2）*强调高阶结构*：从句法分析的一阶 Eisner 算法[36] 到二阶甚至更高阶的最大生成树算法[25] 是这一研究范式的代表。同时，这种思想也激发了以牺牲精确解码换取任意阶特征的非精确解码算法[39]。

全局解码与高阶特征在神经网络兴起之前受到推崇的一大原因是稀疏的词汇化特征表达能力不足。实际，早在 2008 年，Liang 等人就在文献 [92] 中指出解码的复杂程度与表示复杂程度存在某种折中。即简单的解码与复杂的特征表示可以达到复杂的解码与简单的特征表示相同的性能。

Liang 等人的观点在基于神经网络的语言分析中得到充分的验证。基于神经网络的语言分析模型的一大特点就是表示模型无一例外非常复杂。当前性能最优的词性标注模型[6,23] 使用 LSTM 或 CNN 建模词，并使用多层双向 LSTM 建模上下文。性能最优的分词模型[93] 使用也双向 LSTM 建模上下文。性能最优的依存句法分析器[8] 也采用类似的上下文表示。可以说，双向 LSTM 已经成为诸多语言分析模型的标准配置。伴随着 LSTM 的大规模应用，全局解码与高阶特征或被抛弃或被证明对于性能影响较小。Dozat 与 Manning 在 2016 年的文献 [8] 中提出的当前性能最优的深度双仿射句法分析器是这一最新科研范式的最佳例证。深度双仿射句法分析器完全抛弃了稀疏特征时代标配的 Esiner 算法（或更高阶的最大生成树算法），将依存句法分析化简为分类问题。相同的，Ma 等人在 2018 年的文献 [93]

中提出的性能最优的中文分词器也只依赖于字级别的分类，而抛弃了稀疏特征时代的‘‘标配’’条件随机场模型。

基于上述分析，强化基于神经网络的语言分析模型中的表示模型是提高语言分析性能的重要而有效的手段。上下文相关词向量作为一种有效的表示模型有望提高语言分析模型的性能。

1.3 本文的研究内容及章节安排

本文针对上下文相关词向量在语言分析技术中的应用开展一系列研究工作。这些工作按照从词表示，到词法分析，再到句法分析的顺序组织。具体来讲，本文包含 5 章。各章节内容组织如下：

在第1章中，本文介绍了本文课题的研究背景、意义，并对上下文相关词向量及其在语言分析方面的应用现状进行概述与分析，最后对本文主要内容进行了规划。

在第2章中，针对现有上下文相关词向量的效率问题，本文从大部分语言分析模型仅依赖局部上下文表示出发，提出了一种融合相对位置权重的窗口级自注意力机制并用它建模上下文相关词向量。实验结果表明，本文提出的词向量能够在不损失语言分析精度的条件下三倍提升模型的解码速度。

在第3章中，针对语言分析中的切分问题（中文分词与命名实体识别）对合理的片段（词与实体）表示的依赖，本文在上下文相关词向量的基础上提出一种基于简单拼接的片段表示方法并将其应用于半-马尔科夫条件随机场中。在典型切分问题的实验中，本文基于上下文相关词向量的片段表示有效地提高了模型性能。通过进一步融合任务相关的上下文表示以及建模片段级信息的片段向量，本文模型取得了当前最优的性能。本文第3章的工作也经验性地证明了上下文相关词向量可以通过简单组合表示片段。

在第4章中，针对上下文相关词向量对多国语句法分析作用尚无明确结论的现状，本文提出在多国语句法分析中使用上下文相关词向量并在大规模树库上验证上下文相关词向量的有效性。除了获得稳定且显著的性能提升，本文针对提升的原因进行了详细的分析。大量分析实验表明，性能提升的主要原因是上下文相关词向量通过对于未登录词词形的更好的建模有效地提升了未登录词的准确率。基于这一分析结果，本文在资源稀缺的模拟数据中进行了实验，并经验性地证明少量标注数据与上下文相关词向量配合可以获得很好的句法分析性能。

在第5章中，针对使用上下文相关词向量的句法分析参数过多、运行速度较慢的问题，本文提出一种结合探索机制的知识蒸馏算法，以将基于上下文相关词向

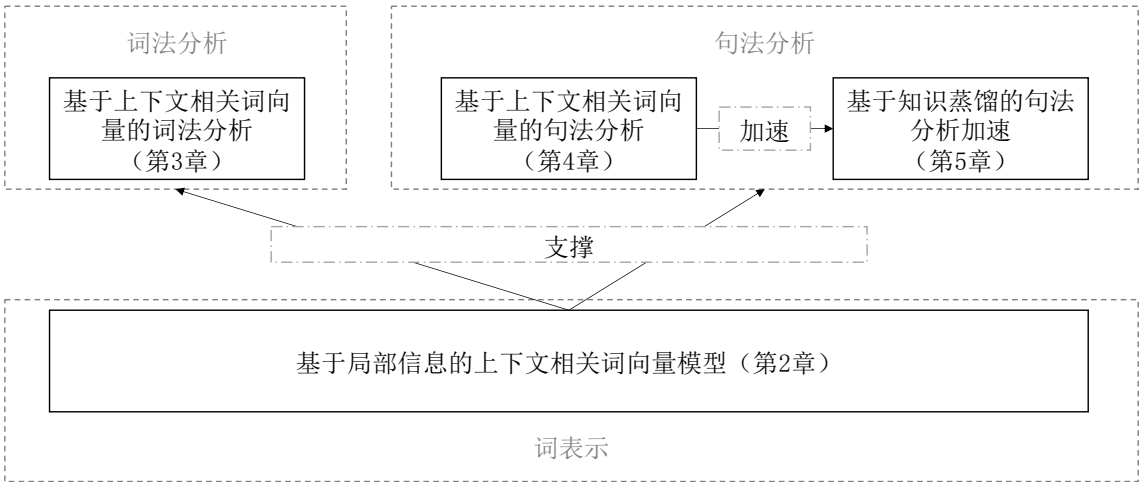

图 1-6 论文总体框架

Fig. 1-6 The organization of the thesis

量的复杂模型用不使用相应词向量的简单模型进行近似，从而在不显著降低性能的情况下提高句法分析速度。实验结果表明，本文提出的方法在损失少量句法分析准确率的情况下，近十倍地提升了速度。

本文各章节之间逻辑关系如图1-6所示。其中，第2章的词表示的工作为后续章节提供支撑，第3章在第2章的基础上研究了词法分析并通过词法分析探索了第2章词表示的组合特性。第4章在第2章的基础上研究了句法分析并详细分析了第2章词表示与句法分析的关系。第5章以为第2章和第4章为基础，将这两章的复杂模型进行速度优化从而使之获得更广泛的应用。

第 2 章 基于局部信息的上下文相关词向量

2.1 引言

上下文相关词向量[12–16] 是静态词向量[1–4] 的一种有效替代。得益于对于相同的词在不同的上下文环境的更好的建模，上下文相关词向量已经给多项句法语义任务带来显著的性能提升。这些任务包括：问答[13,16]、语言角色标注[13]、共指消解[94]、文本蕴含[13,16]、情感分析[13]、短语结构句法[95,96] 以及命名实体识别[16,83]。

上下文相关词向量的成功主要源于对上下文的合理建模。为了合理建模上下文，前人工作往往强调使用深层网络对于整句乃至前后两句进行多次抽象。这种做法使得从大规模文本中学习上下文相关词向量的过程消耗大量的资源。Peters 等人在 2018 年的文献 [13] 中提出使用双向语言模型作为目标学习上下相关词向量的算法—ELMo。他们的模型使用两层 LSTM 表示上下文。根据文献 [13]，在包含 80 万词的 one billion word benchmark 语料（1B word benchmark，文献 [97]）上训练 ELMo 大约需要 1 张 NVIDIA 1080 Ti 显卡两个星期的运算时间。Devlin 等人在 2018 年的文献 [16] 中提出使用 12 层 Transformer 网络建模上下文的模型 — BERT。根据文献 [16]，在 330 万互联网数据上训练 BERT 需要 16 个 TPU 4 天的训练时间。这一训练过程等价于在 8 核 NVIDIA V100 显卡上训练 44 天。① 除了超长的训练时间，这些使用过量参数（over-parameterized）的模型也很大程度地降低了解码速度，使其难于部署到真实世界的应用中。

另一方面，对于像词性标注这样的问题，使用局部上下文可以很好地表示词。在深度学习流行前的时代，使用单词窗②作为 ‘‘特征’’ 表示词的做法是最广泛采用的词表示方法[24,25,29,37]。对于神经网络模型，单词窗模型也在多项语言分析问题中取得了很好的性能[43,98]。一个有趣的研究问题是我们需要对在多大窗口的上下文进行建模？我们能否牺牲一定的全局性换取速度的提升？ Peters 等人在 2018 年的文献 [77] 中尝试将文献 [13] 中的 LSTM 替换为 Gated CNN（GCNN，文献 [78]）以及 Transformer 网络。然而，他们的模型仍强调通过扩大 CNN 的感受域（inception field）或者在整句上使用自注意力机制来表示整句的上下文信息。局部的上下文表示能够达到怎样的效果仍旧没有得到很好的答案。

本章在基于双向语言模型的上下文相关词向量的框架[13] 下探索使用单词窗建

① http://timdettmers.com/2018/10/17/tpus-vs-gpus-for-transformers-bert/

② 即以要表示的词为中心，左右各取若干个词。

模上下文的可能性。基于‘‘不同距离、不同内容的上下文在表示一个词时的相关程度不同’’这一假设，本章提出一种**融合相对位置权重的局部自注意力机制**对局部上下文进行表示。相较文献 [77] 中的 GCNN 和 Transformer，本章模型并不强调表示整句的上下文信息，因而模型更浅、速度更快。与使用 LSTM 的原始 ELMo 相比，这种浅层局部模型在不降低模型性能的前提下获得大约 3 倍的速度提升。

本章在随机采样的 10 万词句英文维基百科数据（记为 enwiki100m）和 1B word benchmark 两个大规模文本语料库中学习了本章提出的上下文相关词向量，并将其应用于五个需要对句子句法特性进行建模的序列标注任务[84] 中。在两个不同数据集的设置下，本章提出的上下文相关词向量均取得了平均意义上与 ELMo 相近的性能，并在几项任务中超越 ELMo。在语义角色标注[75] 和共指消解[76] 的语义实验也证实了本章提出的上下文相关词向量的表示能力与速度优势。

本章的主要贡献包括：

• 本章提出了一种融合相对位置权重的局部自注意力机制对局部上下文进行表示，并利用这种表示学习上下文相关词向量（§2.3）。

• 本章在五项句法任务（§2.4）以及两项语义任务（§2.5）中评价了本章提出的模型的表示能力。句法任务结果显示，本章提出的模型在取得与 ELMo 相同性能的前提下，三倍地提升了速度。语义任务结果显示，本章提出的模型取得了与 ELMo 可比的性能。

• 本章以句法任务为目标，研究了影响局部模型性能的重要因素（§2.4.4）。本章分析显示，使用多层抽象对于上下文的表示能力有重要作用。同时，本章提出的相对位置权重和局部自注意力机制能够相互作用，提高模型表示能力。

本章实验代码开源于：https://github.com/Oneplus/ELMo.

2.2 背景知识

2.2.1 上下文无关词向量

自然语言处理是研究如何对计算机进行编程进而处理和分析自然语言数据的领域。由于词是语言的基本单元，计算机处理和分析自然语言数据的基本而重要的步骤是将表示为符号的词转换为计算机可以处理的数字表示。随着神经网络的发展，研究者们提出了多种将词转换成其低维矢向量表示（即词嵌入）的技术。这些技术背后的基本假设是词义分布式假设[10]，即一个词的词义可以采用它的上下文进行表示。Word2vec[1,2]、FastText[4] 和 GloVe[3] 是这些技术中的代表。Word2vec 通过预测词的上下文来建模分布式假设。FastText 在除了预测上下文，还用子词（subword）的信息建模了词。GloVe 通过预测两个单词的共现来建模分布假设。他

们都在原有词义分布式假设上进一步假设一个词有唯一的向量表示。这一假设使得上下文无关词向量无法建模 ‘‘一词多义’’ 等现象。

2.2.2 上下文相关词向量

与上下文无关词向量不同的是，上下文相关词向量[12–16] 取消了一个词有唯一的向量表示的假设，从而使模型能够根据上下文动态地决定一个词的词义。在这种思路下，一个词的上下文相关词向量 $\mathbf{e}_i$ 可以形式化地定义为

$$\mathbf{e}_i = \mathcal{F}_c(\mathbf{v}_1, \ldots, \mathbf{v}_n)_i 。 \tag{2-1}$$

其中广义向量化函数 $\mathcal{F}_c$ 将一个输入序列映射到相等长度的输出序列中；而 $\mathbf{v}_i \in \mathbf{R}^d$ 是第 i 个词 w_i 的上下文无关的表示。通过根据上下文动态地表示一个词，这个词的 ‘‘一词多义’’ 现象能够得到建模。

2.2.3 基于语言模型的上下文相关词向量

Peters 等人在文献 [13] 中提出通过首先训练双向语言模型然后将语言模型的隐层向量作为上下文相关词向量的方法 — 基于语言模型的上下文相关词向量。本文第1.2.2节已经对于 ELMo 如何建模语言模型，如何建模上下文，以及如何学习参数进行了介绍。本章在基于语言模型的上下文相关词向量的框架下建模本章的上下文相关词向量。正如第1.2.2节讨论的，基于语言模型的上下文相关词向量主要有两个模块，即：

- 词表示函数 ϕ：词表示函数接受一个词 w_i 并且输出它对应的上下文无关词向量 $\mathbf{v}_i$。
- 两个方向的上下文表示函数 $\overrightarrow{\mathcal{F}_c}$、$\overleftarrow{\mathcal{F}_c}$：这两个函数接受一个上下文无关的向量序列，并且输出其上下文相关词向量。

2.3 基于局部信息的上下文相关词向量模型

本章的主要目标是提高上下文相关词向量的训练测试速度。根据文献 [77]，词表示函数 ϕ 输出的上下文无关词向量可以通过预处理与缓存机制快速计算，其效率对于测试速度的影响并不大。因而，本章工作的重点在提出新的上下文表示函数 $\overrightarrow{\mathcal{F}_c}$、$\overleftarrow{\mathcal{F}_c}$ 以获得训练测试的加速（§2.3.2）。但是，提高词表示函数的效率能够给训练带来加速，从而缩短模型开发周期。因此，本章也探索了可以给训练带来潜在加速的词表示函数（§2.3.1）。

对于上下文相关词向量训练的加速问题，除了从词表示与上下文表示的角度出发，Li 等人在 2019 年的文献 [99] 中提出最小化上下文相关词向量与静态词向量距离的训练方式 — SemFit。文献 [99] 使用 SemFit 替代文献 [13] 中的词表级的

对数似然，从而获得了训练速度的提升。但根据文献 [99]，这种方法会带来一定的精度损失。所以本章仍使用文献 [13] 提出的基于双向语言模型对数似然和的学习目标（公式1-7）。

2.3.1 词表示函数

2.3.1.1 基于字 CNN 的词表示方法

文献 [13] 采用 Kim 在 2016 年的文献 [79] 中提出的基于字 CNN 的词表示模型（如图2-1左图所示）。基于字 CNN 的词表示模型通过使用不同宽度的卷积函数在字序列方向滑动建模了不同词片段的重要程度。当然，在卷积函数数量增加时，基于字 CNN 的词表示模型的速度会降低。

2.3.1.2 基于词片段加和的词表示方法

对于建模词片段的问题，基于字 CNN 的方法通过卷积函数自动挖掘词片段（对于‘‘重要程度’’的建模可以认为是一种对于词片段的挖掘）。除了这种自动的方法，另一类常见的方法是使用语言学家定义的词片段作为词的一种切分。从词到词片段序列的过程叫做词*切片*（tokenization）。Luong 等人在 2013 年的文献 [100] 以及 Botha 等人在 2014 年的文献 [101] 中分别证明了切片获得的*词片段的向量*（wordpiece embedding）建模词向量的有效性。为了加速模型学习，本章参考文献 [101] 并使用文献 [16] 中的词切片方法首先获得词片段，然后通过词片段向量加和的方法计算整个词的表示（如图2-1右图所示）。

基于词片段加和的方法加速词表示的原因有两方面。第一方面是由于词切片的输出是不相交的，基于词片段的方法考虑的词片段更少。第二方面是由于加和不包含矩阵乘法，基于词片段的方法运算得更快。但这种表示方法也会相应带来无法建模相交片段，表达能力不足的问题。

2.3.2 上下文表示函数

局部上下文表示　根据上下文表示模型能够建模的上下文的长度，前人工作可以归纳为两类。一类是类似 LSTM 和 Transformer 的对于完整句子的上下文进行建模的模型。另一类是类似 CNN 的对一个窗口内的上下文进行建模的模型（窗口模型）。对于第 i 个词的宽度为 W 的上下文，窗口模型可以形式化地定义为：

$$\mathbf{h}_i = \mathcal{F}_c(\mathbf{v}_1, \ldots, \mathbf{v}_i, \ldots, \mathbf{v}_n) = \mathcal{F}_c(\mathbf{v}_{i-W}, \ldots, \mathbf{v}_i, \ldots, \mathbf{v}_{i+W})。 \tag{2-2}$$

在基于语言模型的上下文相关词向量的框架下，窗口模型可以定义为：

$$\mathbf{h}_i = \overrightarrow{\mathcal{F}_c}(\mathbf{v}_{i-W}, \ldots, \mathbf{v}_i) \oplus \overleftarrow{\mathcal{F}_c}(\mathbf{v}_i, \ldots, \mathbf{v}_{i+W})。 \tag{2-3}$$

需要注意的是，当窗口大小足够大时，窗口模型也可以建模完整的句子信息。

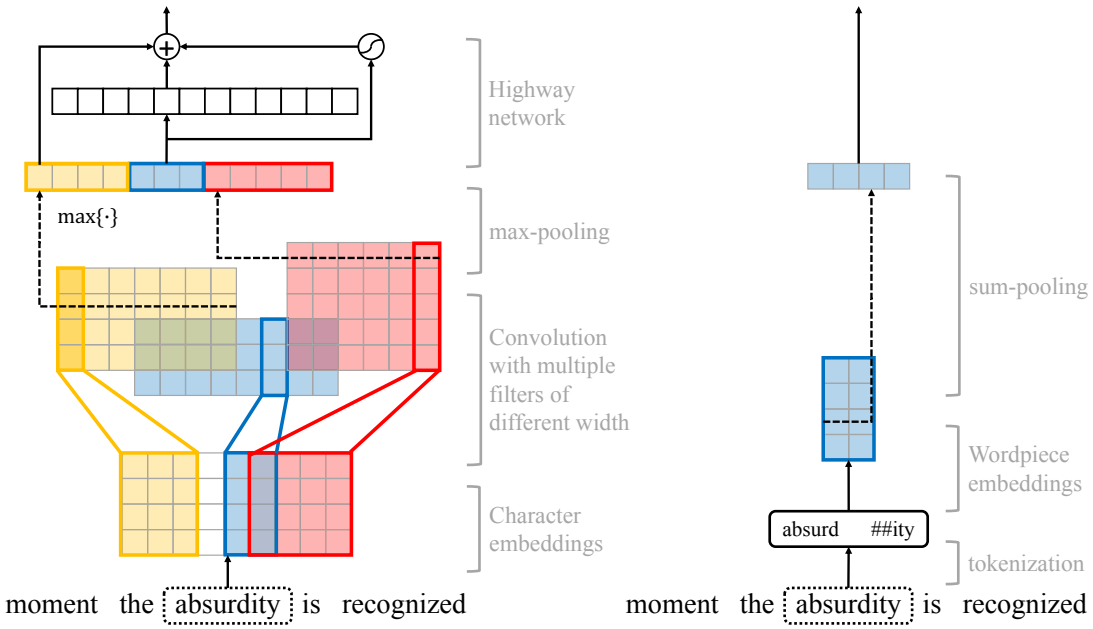

图 2-1 左图为基于字 CNN 的词表示模型，右图为基于词片段加和的词表示模型。

Fig. 2-1 The left figure is the character CNN model and the right figure is the wordpiece summation model.

Bengio 等人在 2003 年的文献 [53] 中首次提出使用神经网络建模语言模型。同时，文献 [53] 也是最早在基于神经网络的语言模型中使用窗口模型表示上下文的工作。具体来讲，为了表示 w_i，文献 [53] 使用拼接窗口内词向量 $\mathbf{v}_{i-W},\ldots,\mathbf{v}_i$，并将其输入前馈神经网络（MLP）的方式建模上下文。以前向语言模型为例，第 i 个词的上文表示 $\overrightarrow{\mathbf{h}}_i$ 可以形式化地描述为：

$$\overrightarrow{\mathbf{h}}_i = \text{linear}\left(\mathbf{v}_{i-W} \oplus \cdots \oplus \mathbf{v}_i\right)。 \tag{2-4}$$

根据前馈神经网络的性质，文献 [53] 可以理解为将窗口内的词向量同时按顺序和维度加权求和。类似的，后向语言模型输出下文表示 $\overleftarrow{\mathbf{h}}_i$ 可以定义为 $\overleftarrow{\mathbf{h}}_i = \text{linear}\left(\mathbf{v}_i \oplus \cdots \oplus \mathbf{v}_{i+W}\right)$。

Minh 等人在 2007 年的文献 [55] 中发展了文献 [53] 的方法并将拼接替换为词向量按照相对距离加权求和。对应的第 i 个词的上文表示 $\overrightarrow{\mathbf{h}}_i$ 可以形式化地描述为：

$$\overrightarrow{\mathbf{h}}_i = \sum_{j=i-W}^{i} c_{i-j} \cdot \mathbf{v}_j。 \tag{2-5}$$

其中，c_{i-j} 是一个标量，可以解释为：根据相对距离决定一个窗口内的词 w_j 与目标词 w_i 的相关程度，一个词距离目标词越近，相关程度越高。类似的，第 i 个词的下文表示 $\overleftarrow{\mathbf{h}}_i$ 可以形式化地描述为 $\overleftarrow{\mathbf{h}}_i = \sum_{j=i}^{i+W} c_{j-i} \cdot \mathbf{v}_j$。

融合相对位置权重的局部自注意力机制 建模一个词与目标词相关程度的做法在自注意力机制（self-attention，文献 [18]）中得到进一步发展。具体来讲，对于一个词 $\mathbf{v}_i$ 及其上文词窗口 $\mathbf{v}_{i-W},\ldots,\mathbf{v}_i$，自注意力机制通过公式：

$$a_{i-j} = \underset{i-W \le j \le i}{\text{softmax}}\left(\frac{\mathbf{v}_i \cdot \mathbf{v}_j^T}{\sqrt{d}}\right) \tag{2-6}$$

计算 w_i 与窗口内每个词 w_j 的权重。其中，d 是 $\mathbf{v}_i$ 的维度。[①] 相较文献 [55] 提出的**相对位置**权重，自注意力机制根据一个词与目标词的**内容**决定其相关程度。直观地讲，内容对于两个词的相关程度的判断有重要作用。以图2-1中的句子为例，在建模 “absurdity’’ 时，“recognized’’ 的相关程度应高于 “the’’。然而，这种方法无法区分不同距离的相同词。因此，在自注意力机制中建模距离信息仍是有意义的。

在文献 [18] 中，词窗口的范围覆盖整个句子。本章旨在通过建模局部信息加速上下文建模，因而只在一个有限范围内计算自注意力。除了使用自注意力分数作为重要程度的衡量，本章也像文献 [55] 一样将相对距离引入模型，使得上下文表示建模了不同距离、不同内容的上下文在表示一个词时的相关程度不同。本章引入相对距离的方式是将其作为一个额外的未归一化的分数与自注意力分数进行加和，得到的上文表示可以形式化地定义为:

$$a_{i-j} = \underset{i-W \le j \le i}{\text{softmax}}\left(\frac{\mathbf{v}_i \cdot \mathbf{v}_j^T}{\sqrt{d}} + c_{i-j}\right)。 \tag{2-7}$$

类似的，下文表示可以定义为：$a_{j-i} = \text{softmax}_{i \le j \le i+W}\left(\frac{\mathbf{v}_i \cdot \mathbf{v}_j^T}{\sqrt{d}} + c_{j-i}\right)$。在计算获得 a_{i-j} 后，第 i 个词的上下文表示可以写作：

$$\overrightarrow{\mathbf{h}^*}_i = \sum_{j=i-W}^{i} a_{i-j} \cdot \mathbf{v}_j。 \tag{2-8}$$

关于位置信息，文献 [18] 提出将建模绝对位置的位置向量（position embeddings）与 $\mathbf{v}_i$ 相加从而使模型捕捉位置信息。相较文献 [18] 的方法，本章方法只依赖相对位置权重 c 表示位置信息。文献 [102] 提出了一种使用相对位置向量的方法表示相对位置。其方法可以形式化地描述为：

$$a_{i-j} = \underset{i-W \le j \le i}{\text{softmax}}\left(\frac{\mathbf{v}_i \cdot \mathbf{v}_j^T + \mathbf{v}_i \cdot \mathbf{c}_{i-j}^T}{\sqrt{d}}\right)。 \tag{2-9}$$

其中，$\mathbf{c}_{i-j}$ 是相对位置向量。相对于文献 [102] 的方法，本章模型对相对位置表示

① 文献 [18] 提出一种将 $\mathbf{v}_i$ 切分为多个子空间并在子空间上计算自注意力权重的方法，即多头自注意力机制（multi-headed attention）。在多头自注意力机制中，d 是每个子空间的维度。本章也使用了多头注意力机制，并将 ‘‘头’’ 数设为 16。

进行了化简。

本章参考文献 [13] 并将窗口内加权平均的词向量$\overrightarrow{\mathbf{h}^*}_i$输入高速公路网络（highway network，文献 [103]）从而获得最终的表示$\overrightarrow{\mathbf{h}'}_i$。综上所述，本章提出的模型在一层一个窗口内的表示可以形式化地定义为：

$$\overrightarrow{\mathbf{h}'}_i = h(\overrightarrow{\mathbf{h}^*}_i) \cdot g(\overrightarrow{\mathbf{h}^*}_i) + \overrightarrow{\mathbf{h}^*}_i \cdot (1 - g(\overrightarrow{\mathbf{h}^*}_i))。 \tag{2-10}$$

其中，$h(\mathbf{x}) = W_h\mathbf{x} + b_h$ 是高速公路网络中的映射函数，$g(\mathbf{x}) = W_g\mathbf{x} + b_g$ 是门控函数。

多层机制 本章参考文献 [77] 使用的多层机制对上下文进行多次抽象（如图2-2所示）。同时，本章层和层之间加入短路机制[74]，从而减少多层带来的梯度消失问题。当定义$\overrightarrow{\mathbf{h}}_i^{(k)}$为第 k 层的第 i 个词的输入时，多层机制可以形式化地定义为：

$$\begin{aligned} \overrightarrow{\mathbf{h}'}_i^{(k)} &= \overrightarrow{\mathcal{F}_c}^{(k)}\left(\overrightarrow{\mathbf{h}}_{i-L}^{(k-1)}, \ldots, \overrightarrow{\mathbf{h}}_i^{(k-1)}\right) \\ \overrightarrow{\mathbf{h}}_i^{(k)} &= \overrightarrow{\mathbf{h}'}_i^{(k)} + \overrightarrow{\mathbf{h}}_i^{(k-1)}。 \end{aligned} \tag{2-11}$$

其中 k 从 0 计数，$k = 0$ 代表上下文无关的词向量。

综合公式2-7到2-11，对于模型的一层，本章提出的融合相对位置权重的局部自注意力机制可以表述为：

$$\begin{aligned} a_{i-j}^{(k)} &= \mathrm{softmax}_{i-W \le j \le i}\left(\frac{\overrightarrow{\mathbf{h}}_i^{(k-1)} \cdot \left(\overrightarrow{\mathbf{h}}_j^{(k-1)}\right)^T}{\sqrt{d}} + c_{i-j}\right) \\ \overrightarrow{\mathbf{h}^*}_i^{(k)} &= \sum_{j=i-W}^{i} a_{i-j}^{(k)} \cdot \overrightarrow{\mathbf{h}}_j^{(k-1)} \\ \overrightarrow{\mathbf{h}'}_i^{(k)} &= h(\overrightarrow{\mathbf{h}^*}_i^{(k)}) \cdot g(\overrightarrow{\mathbf{h}^*}_i^{(k)}) + \overrightarrow{\mathbf{h}^*}_i^{(k)} \cdot (1 - g(\overrightarrow{\mathbf{h}^*}_i^{(k)})) \\ \overrightarrow{\mathbf{h}}_i^{(k)} &= \overrightarrow{\mathbf{h}'}_i^{(k)} + \overrightarrow{\mathbf{h}}_i^{(k-1)}。 \end{aligned} \tag{2-12}$$

2.3.3 实现与模型训练细节

本章在随机采样的 10 万词英文维基百科数据（enwiki100m）以及 1B word benchmark 上训练本章的上下文相关词向量。两个生文本的统计数据如表2-1所示。由表2-1可见，本章使用的生语料句子的单句词数大部分小于 50，所以本章将句子按照 50 的进行了截断。

本章将上下文相关词向量的维度设置为 1,024（前向与后向各 512 维）。为了公平比较，本章也将其他基线系统的维度设置为 1,024。本章根据开发集的实验结果决定模型的层数 L 与窗口宽度 W。如前文所述，本章以双向语言模型的对数似然和（公式1-7）为优化目标训练了本章的模型。本章使用 Adam 算法[104] 优化参

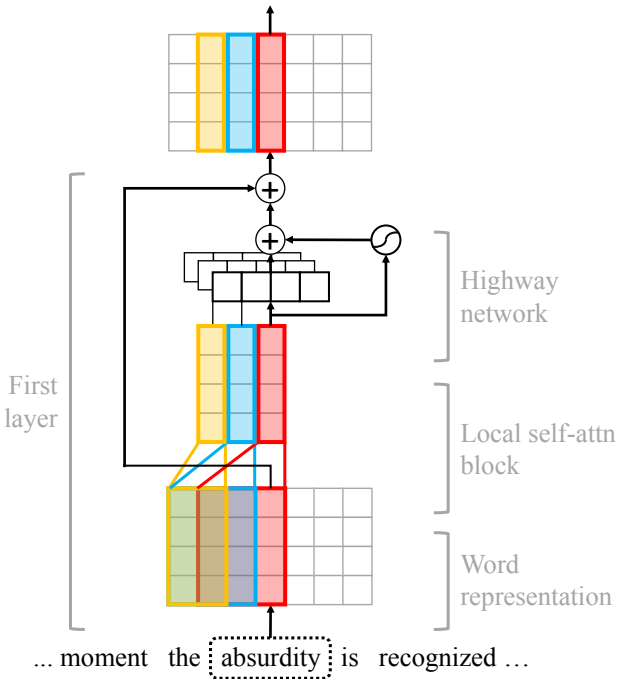

图 2-2 本章使用的多层机制

Fig. 2-2 Our model with multi-layer mechanism

表 2-1 生语料数据统计

Table 2-1 The statistics of raw data

数据集	句子数	小于 50 词的句子比例	词数	词表大小
enwiki100m	6.8M	97.93%	100M	2.8M
1B word benchmark	30.6M	96.62%	776.4M	2.4M

数，并按照文献 [18] 的建议，根据公式

$$\text{rate} = d_{\text{model}}^{0.5} \times \min(\text{step}^{0.5}, \text{step} \times \text{warmup}^{-1.5}) \tag{2-13}$$

调整学习率。[①]

2.3.4 上下文相关词向量的性能评价

本章参考文献 [13][77] 以及 [16] 使用两类下游任务 — 词法-句法任务和语义任务的性能作为上下文相关词向量的性能评价。本章按照公式1-6 使用各隐层的加权平均作为每个词的上下文相关词向量。对于是否精调（fine-tuning）上下文相关词向量，文献 [13] 提出根据开发集性能决定；文献 [16] 则完全使用精调。本章的目标与文献 [77] 最为相似，即通过上下文相关词向量的在具体任务中的迁移性能衡量其表达能力，因而，在所有实验中，本章将本章提出的以及对比的上下文词向量用作固定的词向量，只调整公式1-6中的 γ 与 s_k。

① 这种学习率的调整方法亦名 “noam’’ 算法，$d_{\text{model}} = 512$。

2.4 词法-句法任务评价

2.4.1 设置

数据集与评价指标 本章在一系列词法-句法相关的序列标注任务[84] 上进行了测试。这些任务包含：组块分析（CoNLL00，文献 [105]）、命名实体识别（CoNLL03，文献 [64]）、资源稀缺词性标注（PTB500，文献 [65]）、实体识别（ACE05）以及多风格词性标注（en_ewt，文献 [106]）。本章词法-句法任务评测数据的规模如表2-2所示。

本章关注的组块分析（CoNLL00）、命名实体识别（CoNLL03、ACE05）可以归纳为片段识别问题。本章片段识别问题转化为 BIO 序列标注任务。对于词性标注问题，本章使用标注准确率评价模型性能。对于片段识别问题，本章使用去掉无关片段① 的片段 F1 值评价模型性能。

本章也对解码速度进行了比较。本章根据文献 [77] 评估上下文表示（§2.3.2）的速度，并忽略了词表示（§2.3.1）的时间开销。本章在两种 GPU 架构 — NVIDIA P100 和 NVIDA Titan XP 上评价了解码速度。解码速度采用模型相对 ELMo 的吞吐比率作为模型速度的评价。② 最终速度结果是两种 GPU 架构吞吐率的平均值。

序列标注模型 本章使用可调的词向量与映射后的固定的词向量拼接构成词表示。其中，可调的词向量的维度为 100，映射后的固定词向量的维度也为 100。在获得词表示后，本章参考文献 [84] 将其输入包含 128 维隐含神经元的两层堆叠双向 LSTM[23] 继续编码上下文，并将得到的任务相关的上下文表示输入 CRF 层从而捕捉标签序列的结构信息。

本章参考文献 [84] 并使用比率为 0.33 的 recurrent dropout 机制[107] 训练本章的序列标注模型。Adam 算法也被用于学习序列标注模型参数。根据文献 [84]，神经网络模型的学习过程是非确定性的，且对初始化非常敏感。为了控制初始化对模型的影响，本章采用 5 个不同的随机种子进行实验并报告 5 次实验的结果的平均值和标准差。

2.4.2 基线系统

本章将本章提出的词向量与不同的词向量进行对比，对比的词向量包括：

- GloVe[3]：本章使用在 8 千 400 亿生语料上训练的 300 维 GloVe 词向量作为本章的基线词表示。③

① 通常被标注为 O。

② 在这种设置下，ELMo 的速度为常为 1。

③ http://nlp.stanford.edu/data/glove.840B.300d.zip

表 2-2 数据集规模信息
Table 2-2 The statistics of corpus

数据集	标签数	训练集	开发集	测试集	平均句子长度	小于 50 词的比例
CoNLL00	15	8,936	1,844	2,012	23.72	98.15%
CoNLL03	9	14,041	3,250	3,453	14.53	99.54%
PTB500	41	500	5,527	5,462	23.78	98.36%
en_ewt	17	12,543	2,002	2,077	15.33	98.54%
ACE05	15	10,052	2,421	2,050	14.52	98.86%

• Word2vec[2]：本章在 enwiki100m 以及 1B word benchmark 语料上分别训练 Word2vec 模型，并将其作为基线系统进行比较。为了与本章的 1,024 维上下文相关词向量进行公平比较，本章的 Word2vec 的维度设置为 1,024。

• ELMo[13]：随着句子长度的增加，LSTM 的显存开销会显著增长。分步回传机制（back-propagation through time，BPTT，文献 [108]）通常被用来克服这一问题。分步回传机制将句子按照时序划分为若干块（chunk），并在训练时将前一块的隐层状态与记忆状态输入下一块，而梯度只在块内进行传播。这一技术的关键是记录 LSTM 的状态（stateful），文献 [13] 的实现中应用了这一技术。然而，使用局部上下文表示的模型通常不记录状态（unstateful）。为了公平比较，本章在重新训练的 ELMo 中取消了记录状态的实现方式。

• Transformer[77]：本章提出的模型使用局部自注意力机制。为了与全局自注意力机制进行对比，本章将双向 Transformer 也作为本章基线系统。本章的双向 Transformer 包含 3 层，同时参考文献 [16] 利用（绝对）位置向量作为输入。

• Bengio03[53]：为了与一系列局部上下文表示模型进行比较，本章也使用文献 [53] 提出的基于拼接词向量的表示方法作为基线系统。

• LBL[55]：本章比较的另一个局部上下文表示模型是文献 [55] 提出的使用词向量按照相对位置加权平均的方法。

• GCNN[77]：本章也将本章系统与目前性能最好的局部上下文表示模型 — GCNN 语言模型进行对比。本章 GCNN 包含各 512 个宽度为 {1, 2, 4, 8} 的卷积核。

本章的 ELMo、Transformer、Bengio03、LBL、GCNN 基线模型，以及本章模型（标识为 SelfAttnLBL）都使用基于字 CNN 的词表示（§2.3.1.1）作为上下文无关的词表示输入。

2.4.3 结果

实验结果如表2-3所示。表2-3的第一组显示的是使用开源的上下文无关/相关词向量的序列标注模型的结果。第二组显示的是使用在 enwiki100m 上训练的上下文无关/相关词向量的模型的结果。第三组显示的是在 1B word benchmark 上训练

表 2-3 句法任务实验结果。*ave.* 显示的是各任务性能的宏平均，*SPD* 显示的是速度。

Table 2-3 The results on syntactic problems. The *ave.* column shows the macro-averaged performance. The *SPD* column shows the speed evaluation.

模型	CoNLL00	CoNLL03	PTB500	en_ewt	ACE05	ave.	SPD.
GloVe							
840B.300d	94.71 (±0.08)	89.86 (±0.14)	94.21 (±0.03)	95.99 (±0.10)	83.19 (±0.34)	91.59	-
official ELMo							
small	96.18 (±0.03)	90.25 (±0.31)	95.57 (±0.19)	96.27 (±0.09)	84.80 (±0.42)	92.62	1.51x
medium	96.34 (±0.12)	90.68 (±0.19)	95.70 (±0.09)	96.47 (±0.05)	84.88 (±0.27)	92.81	1.47x
large	96.36 (±0.08)	91.30 (±0.14)	95.68 (±0.22)	96.59 (±0.08)	85.59 (±0.21)	93.11	1.26x
enwiki100m							
Word2Vec	94.53 (±0.11)	87.36 (±0.34)	94.01 (±0.07)	95.64 (±0.08)	82.29 (±0.40)	90.77	-
ELMo	**96.10** (±0.07)	**90.21** (±0.24)	95.60 (±0.17)	**96.75** (±0.09)	84.13 (±0.51)	**92.56**	1.00x
LBL	95.18 (±0.09)	89.35 (±0.38)	95.28 (±0.13)	96.24 (±0.03)	83.23 (±0.28)	91.86	3.59x
Bengio03	95.39 (±0.09)	89.19 (±0.20)	95.51 (±0.11)	96.51 (±0.07)	83.67 (±0.49)	92.05	3.25x
Transformer	95.61 (±0.09)	89.39 (±0.23)	95.41 (±0.31)	96.35 (±0.11)	83.32 (±0.26)	92.01	**3.96x**
Gated CNN	95.30 (±0.17)	89.35 (±0.17)	95.41 (±0.24)	96.38 (±0.08)	83.40 (±0.28)	91.97	3.72x
SelfAttnLBL	95.85 (±0.03)	90.11 (±0.23)	**95.79** (±0.14)	96.60 (±0.07)	**84.24** (±0.26)	92.52	3.15x
1B word benchmark							
Word2Vec	94.71 (±0.12)	88.71 (±0.41)	94.44 (±0.06)	95.71 (±0.09)	82.51 (±0.32)	91.22	-
ELMo	**96.40 (±0.05)**	**90.83 (±0.25)**	95.59 (±0.16)	96.53 (±0.06)	**84.84 (±0.20)**	**92.84**	1.00x
SelfAttnLBL	96.12 (±0.09)	90.62 (±0.24)	**95.96 (±0.12)**	**96.54 (±0.09)**	84.76 (±0.28)	92.80	3.14x
Previous SOTA							
	文献 [109]	文献 [13]	-	文献 [110]	文献 [111]		
	96.72	92.22 (±0.10)	-	95.82	83.6		

词向量对应的结果。[①] 最后一组显示的是几个数据集上对应的当前最优模型的结果。根据第二组和第三组中 Word2vec 与其他上下文相关词向量的对比，使用上下文相关词向量做输入的序列标注模型的性能显著好于使用上下文无关词向量的模型。这一观察证明了上下文相关词向量的有效性。由于这些上下文相关词向量都是基于语言模型的构建的，这一结果也间接地验证了文献 [77] 关于语言模型对于学习上下文相关词向量效果的假设。

对于 enwiki100m 上的实验（表2-3第二组），本章提出的上下文相关词向量给模型带来了所有基线模型中第二大的提升。相较提升最大的 ELMo，本章提出的词向量只落后 0.04 分的平均分。对于在 1B word benchmark 上（表2-3第三组），本章提出的词向量显示出于 enwiki100m 相同的趋势。与 ELMo 的差距也是 0.04。根据最后一列的速度对比，在不损失准确性的前提下，本章提出的词向量带来了 3 倍的速度提升。

① 由于在 1B word benchmark 上训练上下文相关词向量需要过长的时间，本章根据表2-3中第二组的结果只在 1B word benchmark 上比较了本章提出的 SelfAttnLBL 与 ELMo 的性能，而忽略了其他基线系统。

通过与其他基于局部上下文表示的词向量（Bengio03、LBL 以及 GCNN），本章提出的方法性能更好，而且速度相当。在与 Transformer 的对比中，本章提出的词表示平均高出 0.5。本章将在后文分析相对 Transformer 性能提升的主要原因。

在 1B word benchmark 数据集上训练的词向量可以与官方 ELMo[13] 进行合理对比。相比官方 ELMo，本章重现的 ELMo 在评价性能上有 0.3 个点的差距。本章认为这一差距一方面来源于是否使用分步回传机制，另一方面在于训练算法的不同。① 相较使用性能相近的中等 ELMo 模型（表2-3中 medium 行），本章的提出的算法运行速度更快。

同时，本章也在表2-3中前人最优的结果进行了对比。本章提出的词向量在 *en_ewt*、*ACE05* 以及 *CoNLL00* 上分别取得了与超越或基本持平前人最优结果的性能。这一比较反映了本章提出方法的有效性。本章提出的词向量在 *CoNLL03* 上相较当前最优模型差距较大。本章认为这一差距是由不同的序列标注方法导致的。

2.4.4 分析

2.4.4.1 超参分析

如表2-3所示，本章提出的词向量取得了与 ELMo 相近的性能。本章同时关注模型中哪些部分对词向量表示能力起关键作用。如前文所述，本章提出的模型主要包含两种参数：1）窗口大小；2）层数。在这一章中，本章将研究这些超参对于词向量性能的影响。分析这些参数的影响可以从侧面反映出建模局部上下文时的重要因素。

窗口大小的影响　本章首先研究了窗口大小的影响。本章在两层和三层 SelfAttnLBL 模型上分别测试了包括 2、4、8、16、32 在内的 5 种窗口大小。根据表2-2，本章关注的大部分词法-句法任务的句子长度都小于 50，其中 3 层 16 宽、2 层 32 宽以及 3 层 32 宽都可以对大部分完整句子进行建模。开发集上的实验结果如图2-3左图所示。根据图2-3，在两层和三层模型中增大窗口大小到 16 时可以获得显著的性能提升，继续增大到 32 时产生性能的下降。这一结果显示对于局部表示模型来讲，并非窗口越大越好。过大的窗口迫使自注意力网络判断更多的上下文的重要程度，从而增加模型的学习难度。

由于不同的模型层数与不同窗口大小的组合会导致不同的 ‘‘感受域’’ 大小。② 本章同时以 ‘‘感受域’’ 大小为指标重新绘制了窗口上述实验。其结果显示于图2-3右图中。根据图2-3右图，感受域大小在 24 到 48 之间开发集性能较好，但三层模

① 官方 ELMo 使用 AdaGrad 算法进行训练。

② 即在窗口模型中实际可以关注到的输入大小。举例来讲，窗口为 4 的三层模型的感受域大小为 12。

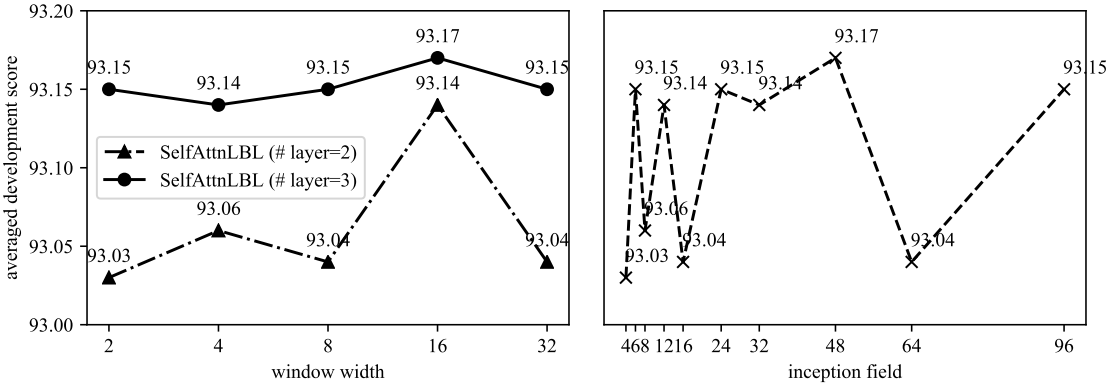

图 2-3 左图显示两种模型中窗口大小对于模型性能的影响。右图以 ‘‘感受域’’ 大小为横轴重新绘制了左图结果。

Fig. 2-3 The effect window size. The x-axis in the right figure is the size of inception field taking multi-layer mechanism into consideration.

型的整体性能较好，这说明相对感受域，模型性能的作用受到模型复杂程度的影响更大。文献 [77] 强调了多层模型在建模上下文相关词向量时的抽象能力。文本将上述实验观察归因于多层模型的更强的抽象能力。

同时，这一实验结果表明窗口大小为 16 的设置具备较好的开发集性能。本章以此为基础进行后续分析实验。

层数的影响 接下来，本章对层数的影响进行了研究。根据前文，本章设定窗口大小为 16，并研究了 1、2、3、4 层模型的性能。其中实验结果如图2-4所示。根据这一结果，层数为 1 的词向量性能显著低于其他层数。同时，随着层数增加为 4，本章观察到轻微的开发集性能下降。出于简化模型的角度考虑，本章设定模型层数为 3。

2.4.4.2 消融实验

局部模型于相对位置权重的影响 根据表2-3，本章提出的词向量相对 Transformer 获得了平均分 0.5 的性能提升。由于本章模型与 Transformer 的主要差别在于：1）局部自注意力机制；2）建模相对位置。为了研究模型性能提升的关键，本章在这一节进行了消融实验。实验的系统除了 LBL、Transformer 和 SelfAttnLBL，还包括：

- 只包含局部注意力机制的模型：SelfAttn；
- 包含局部注意力机制与绝对位置向量的模型：SelfAttn+position embeddings。

这一结果显示在表2-4中。由于 Transformer 与 SelfAttn+position embeddings 的差别在于建模信息的局部/全局性，通过对比表2-4中对应的行，本章认为在上下文相关词向量中建模局部信息的可以取得与全局信息相近甚至更好的效果。由于 SelfAttn+LBL 与 SelfAttn+position embeddings 的主要区别在于对使用绝对位置向量/相

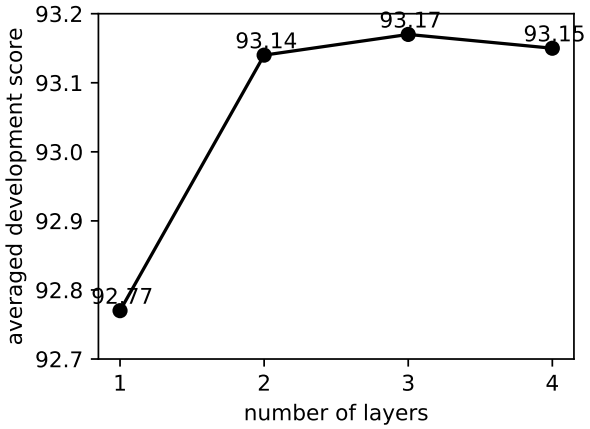

图 2-4 模型层数对于性能的影响
Fig. 2-4 The effect of number of layers

表 2-4 消融实验结果
Table 2-4 The ablation results

注意力	位置	模型	开发集	测试集
全局注意力	绝对位置	Transformer	92.77	92.01
-	相对位置	LBL	92.57	91.86
局部注意力	-	SelfAttn	92.63	92.01
局部注意力	绝对位置	+position embeddings	92.78	92.16
局部注意力	相对位置	+LBL	**93.17**	**92.52**

对位置权重，通过对比表2-4中对应的行，本章提出的融合相对位置权重的局部自注意力模型取得了明显的性能提升。这表明使用相对位置权重的重要性。

综合上述消融实验结果，本章提出的 SelfAttnLBL 带来的性能提升不是局部自注意力与相对位置权重性能的简单累加，二者通过相互作用达到更好的效果。

输入表示的影响 如第2.3.1.2节所述，本章尝试通过加速词表示获得训练速度的提升。在这一节中，本章将字级别 CNN 的词表示替换为本章第2.3.1.2 提出的基于词片段加和进行词表示的模型。其实验结果如表2-5所示。这一结果显示字级别 CNN 相较词片段表示加和的方式在表示能力上仍有优势。本章将这一观察归因为词片段依赖语言学家定义的形态学规则，不能建模相交词片段，同时片段向量加和的表示能力不足。

2.4.4.3 未标注数据规模的影响

本章分析了未标注数据规模对模型性能的影响。本章选取了 1、2、5、10、80 万为不同的数据规模。其中，1、2、5、10 万数据是从 enwiki100m 中采样获得。80 万数据从 1B word benchmark 中获得。实验结果如图2-5所示。图中包括三种模型：

表 2-5 不同词表示模型的效果。

Table 2-5 The effect of different word representations.

模型	开发集	测试集
SelfAttnLBL + CNN	**93.17**	**92.52**
SelfAttnLBL + wordpiece sum	92.87	92.00

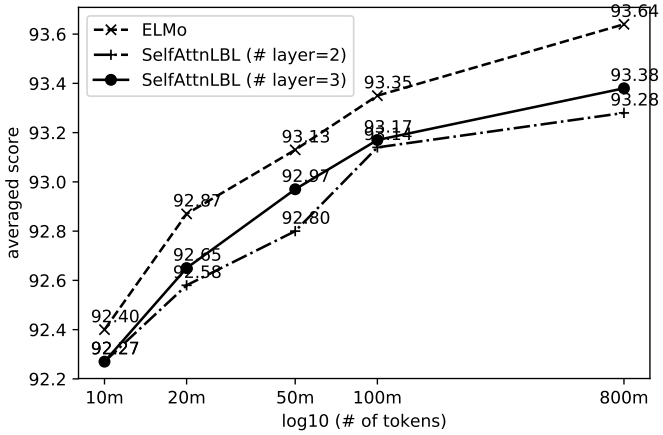

图 2-5 未标注数据规模对于模型性能影响

Fig. 2-5 The effect of unlabeled data size

ELMo，两层 SelfAttnLBL 以及三层 SelfAttnLBL。图2-5的结果显示随着未标注数据量的增加，上下文相关词向量的表示能力逐步提高。这一观察符合预期，即使用更多的训练数据能够提高学得的上下文相关词向量的性能。

2.4.4.4 语言模型性能与表示能力的关系

最后，本章分析了语言模型性能与上下文相关词向量表示能力的关系。本章以模型训练最后一轮训练集的困惑度（perplexity）作为语言模型性能的评价。该结果显示于图2-6中。根据图2-6左图，困惑度与模型性能呈现出相关性，即困惑度越小模型性能越好，但两者相关性并不显著。① 可见，通过网络结构提高语言模型拟合训练数据的能力并不能保证提高对应上下文相关词向量的表示能力。但根据图2-6右图，对于相同的语言模型，降低训练的困惑度能够带来性能的提升。②

2.5 语义任务评价

本章参考文献 [77] 并在两个语义任务 — 语义角色标注[75] 和共指消解[76] 上验证本章提出的上下文相关词向量的有效性。下文将对这两个任务及模型进行介绍。

① 两者的皮尔逊相关系数为-0.7211，p-value>0.1。

② 两条曲线的皮尔逊相关系数为：-0.7801 和-0.9633。p-value 均小于 0.01。

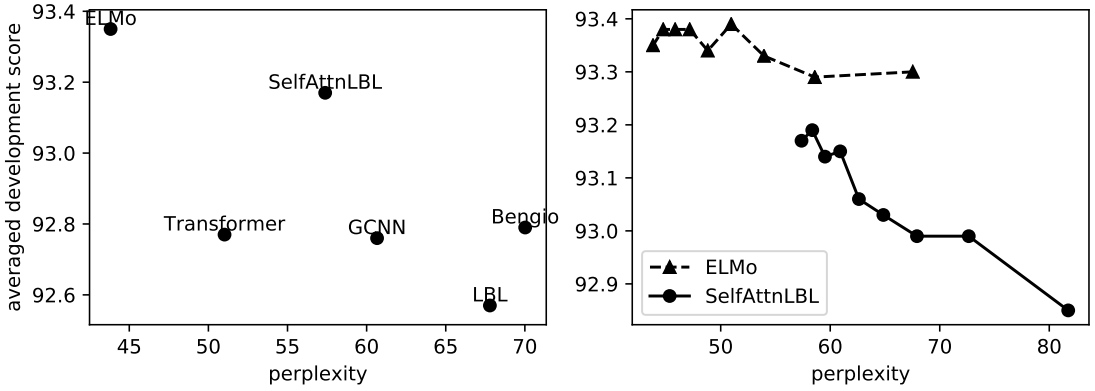

图 2-6 模型的训练困惑度与下游任务性能之间的关系。左图显示不同语言模型的结果，右图显示两种语言模型不同迭代轮次的结果。

Fig. 2-6 The relation between training perplexity and model's performance. The left figure shows the perplexities of different models and the right figure shows the perplexities of one certain model at different iteration.

表 2-6 语义任务结果

Table 2-6 The results of semantic evaluation

模型	语义角色标注	共指消解
GloVe	79.00 (±0.11)	61.60 (±0.06)
ELMo	**82.81** (±0.12)	**65.67** (±0.27)
SelfAttnLBL	82.03 (±0.15)	65.09 (±0.18)

2.5.1 模型

语义角色标注 语义角色标注系统要从句子中找出谓词-论元结构。谓词-论元结构可以理解为回答句子中的“谁对谁做了什么，怎么做的”的问题。本章在 OntoNotes 数据集[75] 上进行了实验，使用的语义角色标注模型为 He 等人在 2017 年的文献 [112] 提出的基于 8 层 BiLSTM 序列标注的语义角色标注模型。本章基于 AllenNLP[113] 实现了文献 [112] 中的模型。本章使用片段 F1 值作为语义角色标注的评价指标。与句法评测类似，本章使用 5 个不同随机种子做初始化，并报告结果的均值与方差。

共指消解 共指消解是将文中提及的实体聚类到现实世界实体的任务。本章在 CoNLL-2012 数据集[76] 上进行验证。本章采用 Lee 等人在 2017 年文献 [114] 中提出的使用 BiLSTM 与注意力机制的端到端共指消解模型。与语义角色标注类似，本章基于 AllenNLP 实现文献 [114] 的模型并相应地汇报了 5 个不同随机种子做初始化的实验结果的均值方差。

2.5.2 结果

本章对比了使用 GloVe 做静态词向量以及使用 ELMo 的模型。表2-6显示了本章语义评价的结果。根据表2-6，相对 GloVe 的结果，使用上下文相关词向量能够给两个语义任务带来显著的性能提升。对比 ELMo 结果，本章模型在语义角色标注上有 0.78 的差距，在共指消解上有 0.58 的差距。可见在语义任务上使用建模全局信息的模型仍有其意义。但本章提出的模型主要针对需要建模局部信息的句法问题，并且相对 ELMo 有三倍的速度优势，可以认为其仍有较大的意义。

2.6 本章小结

本章针对上下文相关词向量的效率问题，根据语言分析主要依赖词窗口的特性，提出一种融合相对位置权重的局部自注意力机制建模上下文相关词向量。并在五项句法任务和两项语义任务上验证本章上下文相关词向量的有效性。实验结果表明，所提出的词向量能够在不损失语言分析精度的条件下三倍提升模型的解码速度。

第 3 章 基于上下文相关词向量的词法分析

3.1 引言

切分问题是识别输入序列的子序列并给其分配标签的问题。包括命名实体识别[115]、意见提取[116] 和中文分词[117] 在内的诸多多自然语言处理任务可以建模为切分问题。合理地表示切分中的**片段**对于切分的性能至关重要。切分问题可以通过给每个输入单元（例如单词或字符）标注一个代表边界的标签转换为序列标注问题。条件随机场[91] 等一类序列标注模型都可以用来建模这类问题。相较序列标注模型，直接建模片段的模型能够有效利用片段信息，不受标签的级联错误的影响。半-马尔科夫条件随机场（Semi-Markov CRF，简称 semi-CRF，文献 [118]）是一种能直接建模片段的模型。semi-CRF 模型显示地建模了输入序列片段（半-马尔科夫链①）的条件概率。这使得 semi-CRF 能自然地建模切分问题。

然而，为了取得较好的切分性能，传统的 semi-CRF 模型很大程度地依赖专家定义的表示片段的特征。近年来，神经网络模型在自然语言处理中取得了很大的成功。神经网络建模自然语言结构的组合特性以及从大规模未标注数据中学习表示的能力是这些成功的关键。在 semi-CRF 中使用神经网络表示片段是一个有趣的研究方向。一方面，诸如 LSTM[17] 等网络结构可以通过组合输入向量的方式建模片段的；另一方面，诸如 Word2vec[2] 一类的词向量算法可以从大规模未标注数据中学习片段的整体表示。

在使用神经网络表示片段这方面，前人工作[119,120] 主要探索了使用不同网络将输入向量组合为片段表示，并未对影响模型性能的关键部分进行探索，也没有关注如何将片段整体进行表示并用作计算。本章将神经网络模型与 semi-CRF 进行结合，并且系统地研究了 semi-CRF 中使用神经网络表示片段的问题。本章研究了对于输入进行上下文表示的作用，同时本章研究了多种组合方式以及如何从生文本中学习一个片段整体的表示。本章在一组典型切分问题 — 组块识别（chunking）、命名实体识别（NER）与中文分词（CWS）上进行了实验。结果表明，建模上下文对于模型的最终性能有显著影响。本章提出的输入拼接可以获得与 SRNN 相似的性能，并三倍地提升解码速度。这一系列使用 semi-CRF 的模型都好于序列标注和 CRF 模型，可见建模片段的有效性。

在此基础上，本章提出使用上下文相关词向量分别代替输入向量以及进行上

① 半-马尔科夫链的状态代表输入序列的一个片段形成一个整体单元

下文表示后的向量以讨论上下文相关词向量能否通过直接的简单组合进行片段表示。结果表明，上下文相关词向量可以通过简单组合取得相较对静态词向量进行上下文表示模型更好的性能，然而在上下文相关词向量的基础上进一步进行任务相关的上下文表示可以带来更大的提升。同时，本章使用上下文相关词向量的 semi-CRF 相对序列标注模型取得 13.15% 的相对错误率降低，进一步证明了建模片段的有效性。

本章的主要贡献包括：

• 本章全面地研究了在 semi-CRF 中使用神经网络表示片段的问题。本章研究了多种通过输入单元表示片段的方法以及将片段整体进行表示的方法（§3.4.1）。这种方法给模型带来稳定的性能提升（§3.4.2）。

• 本章对 semi-CRF 进行了充分的实验（§3.6）。这些实验研究了上下文表示以及片段表示对于模型性能的影响。本章提出的模型相对当前最优的序列标注模型取得了显著的性能提升。

• 本章以 semi-CRF 为工具研究了使用上下文相关词向量组合表示片段能力（§3.5），并经验性地验证上下文词向量的组合能力以及在其基础上加入任务相关上下文表示的有效性。

本章代码开源于：https://github.com/Oneplus/semiCRF.

3.2 问题定义

图3-1 展示了一个命名实体识别与一个中文分词的例子。对于命名实体识别的例子，词的切分序列（*“Michael Jordan”: PER, “is”: NONE, “a”: NONE, “professor”: NONE, “at”: NONE, “Berkeley”: ORG*）代表 “Michaels Jordan’’ 这个片段是一个人名，“Berkeley’’ 这个片段是组织名。在命名实体识别的例子中，字的切分序列（‘‘浦东”, ‘‘开发”, ‘‘与”, ‘‘建设’’）代表识别出的词。这些列子中，切分任务接受一个输入序列，并将其切分为若干不相交的子序列。

切分问题可以形式化地定义为，给定长度为 n 的输入序列 $\mathbf{x} = (x_1, \ldots, x_n)$。定义 $x_{a:b}$ 代表 $\mathbf{x}$ 的子片段 $(x_a, \ldots, x_b)$。$\mathbf{x}$ 的片段可以定义为一个三元组 (u, v, y)。这个三元组代表子序列 $x_{u:v}$ 被标注为 y 标签。输入 $\mathbf{x}$ 的切分是 $\mathbf{x}$ 的片段序列 $\mathbf{s} = (s_1, \ldots, s_p)$。其中，连续两个片段相邻接，即当 $s_j = (u_j, v_j, y_j)$ 时，$u_{j+1} = v_j + 1$。对于给定输入句子 $\mathbf{x}$，切分问题可以定义为找到 $\mathbf{x}$ 最可能的片段序列 $\mathbf{s}$ 的问题。

3.3 基于半-马尔科夫条件随机场的词法分析

半-马尔科夫条件随机场模型（Semi-Markov CRF，简称 semi-CRF，如图3-2所

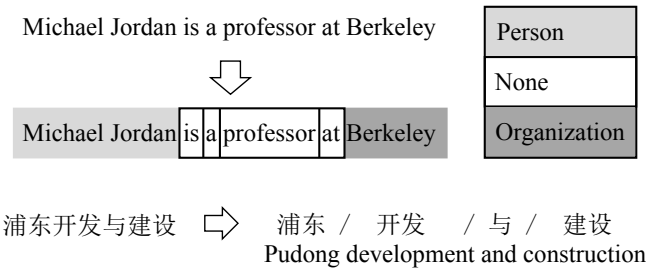

图 3-1 命名实体识别（上图）与中文分词模型（下图）的例子。

Fig. 3-1 Examples for named entity recognition (above) and Chinese word segmentation (below).

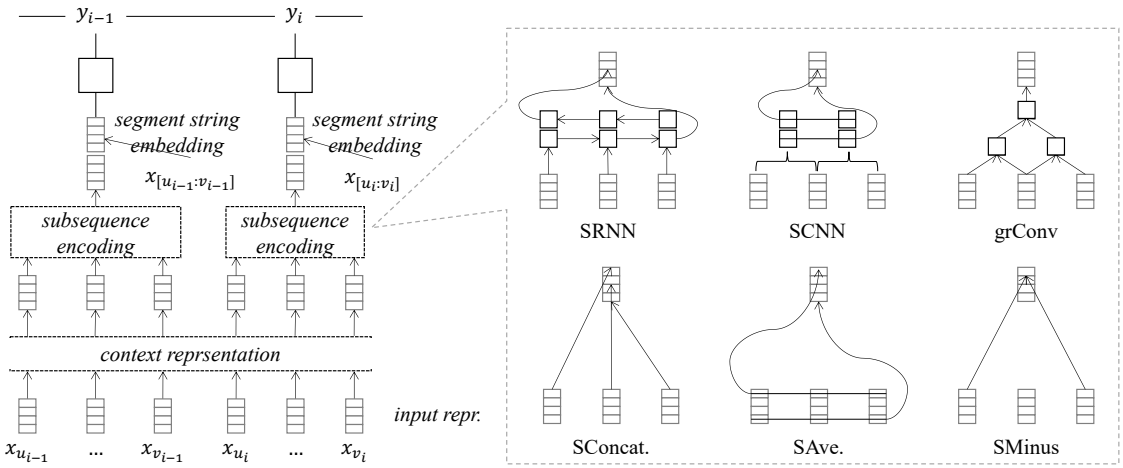

图 3-2 本章提出的融合输入组合函数与片段表示的 semi-CRF 模型。同时，改图显示了本章研究的输入组合函数 — SRNN、grRecNN、SAve、SCNN、SConcat 以及 SMinus。

Fig. 3-2 An illustration for the framework of our neural semi-CRF model, along with the composition functions: SRNN, grRecNN, and our SAve (Eq. 3-5), SCNN (Eq. 3-6), SConcat (Eq 3-7) and SMinus (Eq. 3-8).

示，文献 [118]) 是对于给定 **x** 产生 **s** 的条件概率的一种建模

$$p(\mathbf{s} \mid \mathbf{x}) = \underset{\mathbf{s}' \in \mathbf{S}}{\text{softmax}}\,(W \cdot G(\mathbf{x}, \mathbf{s}))。 \tag{3-1}$$

其中，$\mathbf{S}$ 是所有可能的片段序列，$G(\mathbf{x}, \mathbf{s})$ 是特征函数。$G(\mathbf{x}, \mathbf{s})$ 将输入 **x** 与片段序列 **s** 转化为或稀疏或稠密的向量表示。W 是对应的权重向量。通过 $W \cdot G(\mathbf{x}, \mathbf{s})$，输入 **x** 与片段序列 **s** 共同出现的可能性得到计算。

如果限制特征函数只关注每个片段自身，而不考虑片段和片段之间的高阶关系，$G(\mathbf{x}, \mathbf{s})$ 可以被分解为一系列定义在单个片段上的特征函数 $g(\mathbf{x}, s_j)$ 的加和，即 $G(\mathbf{x}, \mathbf{s}) = \sum_{j=1}^{p} g(\mathbf{x}, s_j)$。这一分解使得最优解序列以及边缘概率等解码问题可以通过动态规划进行计算。对于 0-阶 semi-CRF，为了计算其最优序列，可以定义动态规划状态 α_j 为以第 j 个词结尾的最优的解码序列。同时 α_j 可以采用如下公式递推计算

$$\alpha_j = \max_{l=1...L,y} \Psi(j-l,j,y) + \alpha_{j-l-1}。 \quad (3\text{-}2)$$

其中 L 是一个人工定义的片段的最大长度。$\Psi(j-l,j,y)$ 是代表形成片段 $s=(j-l,j,y)$ 的分数。[①] $\Psi(j-l,j,y)$ 可以通过 $\Psi(j-l,j,y)=W\cdot g(\mathbf{x},s)$ 计算。

前人工作[115–118] 在使用 semi-CRF 模型时往往将 $g(\mathbf{x},s)$ 建模为一个稀疏的 0-1 向量。其中每一个为代表 $\mathbf{x},s$ 是否符合某些预定义的特征。一般来讲，这类特征包括两类：1）输入单元级特征，例如 ‘‘在 i 位置的词的词形’’；2）片段级特征，例如 ‘‘片段的长度’’。

Kong 等人在文献 [119] 中提出使用 RNN 将输入单元的向量组合成片段表示的片段循环神经网络模型（SRNN，模型结构参考图3-2）。文献 [119] 首次将 semi-CRF 与神经网络进行了结合。SRNN 模型使用双向 LSTM 建模 $g(\mathbf{x},s)$。SRNN 首先用一个双向 LSTM 对于输入序列 $\mathbf{x}$ 的上下文进行建模，并将 BiLSTM 的每个位置的隐层输出作为对应词的表示 $\mathbf{h}$。对于一个片段 $s_j=(u_j,v_j,y_j)$，其输入单元子序列 $(x_{u_j},\ldots,x_{v_j})$ 的隐层表示 $(\mathbf{h}_{u_j},\ldots,\mathbf{h}_{v_j})$ 接下来输入另一个双向 LSTM，并取出前向 LSTM 的最后一个隐层输出和后向 LSTM 的第一个输出拼接通过非线性激活函数 ReLU 最终获得 $g(\mathbf{x},s)$。Kong 等人[119] 率先研究了 semi-CRF 与神经网络的结合。双向 LSTM 可以看做是一种 ‘‘神经网络化’’ 的输入单元级特征。Zhuo 等人在 2016 年的文献 [120] 提出另一种网络结构 — 带门控的递归网络（gated recursive neural network，简称 grRecNN，参考图3-2）。这种网络递归地将输入归纳为表示。

Kong 等人的工作以及 Zhuo 等人的工作都显示合理的片段表示能够提高分割任务的性能。然而，他们的工作仅对使用 RNN 建模片段进行了探索，忽略了其他潜在的网络结构。除此之外，前人研究证明片段级的特征在 semi-CRF 中非常有效，他们的工作也没有对这类特征进行探索。

3.4 半-马尔科夫条件随机场中的片段表示

对于 semi-CRF 来讲，片段表示是影响性能的关键。本章从两个角度研究了 semi-CRF 中的片段表示的问题。这两个角度包括：1）将片段内的输入单元组合为片段表示；2）将片段作为整体通过片段向量的转化为片段表示。

对于采用组合输入单元的片段表示模型，其主要包括三个组成部分：1）输入表示模块（§3.4.1.1）：这一模块将输入 $\mathbf{x}=(x_1,\ldots,x_n)$ 转换为对应的上下文无关的表示 $(\mathbf{v}_1,\ldots,\mathbf{v}_n)$；2）上下文表示模块（§3.4.1.2）：这一模块将上下文无关表示转化为上下文相关表示 $(\mathbf{h}_1,\ldots,\mathbf{h}_n)$；3）片段表示模块（§3.4.1.3）：这一模块将一个片

① 即动态规划语境下的转移分数

段 $s = (u, v, y)$ 内的上下文相关表示 $(\mathbf{h}_u, \ldots, \mathbf{h}_v)$ 转化为片段表示。本章尝试了不同的输入、上下文以及片段表示方式。

对于采用片段向量进行片段表示的算法，本章将一个片段 (u, v, y) 在句子中的子串 $x_{[u:v]}$ 通过查表的方法获得其向量表示。这一方法借鉴了前人词向量的工作[2,3]。本章主要研究了如何构造片段以及如何从生语料中学习其向量。本章在第3.4.2节中研究了不同的构造片段的方法。

3.4.1 通过输入单元组合网络表示片段

3.4.1.1 输入表示

词向量是自然语言处理的基础。本章将词向量形式化地定义为 ϕ。ϕ 存储了从词到其向量表示的映射关系。

3.4.1.2 上下文表示

文献 [119] 与文献 [120] 的一个主要的区别在于上下文表示方法的不同。文献 [119] 采用双向 LSTM 编码上下文而文献 [120] 只使用输入表示直接作为上下文表示。在基于神经网络的 semi-CRF 中，两种方法孰优孰劣并没有明确的结论。本章分别尝试了两种上下文表示的方法，即使用 LSTM 的方法

$$(\mathbf{h}_1, \ldots, \mathbf{h}_n) = \text{BiLSTM}(\mathbf{v}_1, \ldots, \mathbf{v}_n)。 \tag{3-3}$$

以及直接使用输入表示作为上下文表示的方法

$$(\mathbf{h}_1, \ldots, \mathbf{h}_n) = (\mathbf{v}_1, \ldots, \mathbf{v}_n)。 \tag{3-4}$$

3.4.1.3 基于组合的片段表示

在获得上下文表示 $\mathbf{e}$ 后，本章使用一个神经网络将片段 (u, v, y) 的表示子序列 $(e_u, \ldots, e_v)$ 转换为一个固定长度的片段表示。由于片段具有变长的特性，这种网络需要具备将任意长度输入序列转换为向量的能力。文献 [119] 使用 RNN 处理输入变长的输入数据；文献 [120] 则使用递归神经网络完成类似的工呢（参考图3-2）。在接下来的章节中，本章提出四种能够处理变长输入的神经网络模型。

片段池化网络（Segmental Average，SAve） 基于神经网络的词袋模型（Neural Bag-of-Word，简称 NBOW，文献 [121]）通过将一个变长序列的表示进行加和从而获得这个序列的向量表示。Cai 与 Zhao 在 2016 年的文献 [85] 中将 NBOW 应用于中文分词中的词表示并观察到性能的提升。本章也尝试了类似的思路。具体来讲，本章使用平均池化网络表示了片段 $s^{(\text{comp})}$，即

$$s^{(\text{comp})} = \frac{1}{v-u+1}\sum_{i=u}^{v}\mathbf{h}_i。 \tag{3-5}$$

本章使用平均池化机制的以期能够缓解由于输入数量不同导致的数值范围不同的问题。

片段卷积网络（Segmental Convolutional Neural Network，SCNN） *SAve* 中的池化机制会忽略片段内部输入和输入之间的顺序关系。但这种位置关系在识别片段时往往是非常重要的。卷积神经网络（convolutional neural network，CNN，文献[43]）能够编码片段内输入的 n-gram 信息，所以可以从一定程度上缓解这一问题。本章使用时序 CNN 建模片段。时序 CNN 在输入单元方向上滑动若干卷积函数，并使用池化函数对卷积函数收集到的表示进行运算。这种方法可以形式化地定义为：

$$s^{(\text{comp})} = \text{Conv}(\mathbf{h}_u, \ldots, \mathbf{h}_v)。 \tag{3-6}$$

在实践中，在片段内使用卷积函数是非常耗时的。本章采用一种折中方案，即在整个句子表示上滑动卷积函数，但在片段内部进行池化，从而加速了片段表示。

片段拼接网络（Segmental Concatenation，SConcat） 为了能够完整地保留片段内输入的顺序信息，本章提出使用拼接网络进行片段表示。为了使拼接网络能够建模变长输入，本章利用 semi-CRF 在解码过程中需要设定最大片段长度 L 的特性，使用补零（padding）机制将变长输入转换为定长表示。这一过程可以形式化地定义为：

$$s^{(\text{comp})} = W^{(\text{sconcat})}(\mathbf{h}_u \oplus \cdots \oplus \mathbf{h}_v \oplus \underbrace{0 \oplus \cdots \oplus 0}_{L-(v-u+1)\text{ zeros}}), \tag{3-7}$$

其中 $\oplus$ 代表向量拼接。$W^{(\text{sconcat})} \in \mathcal{R}^{(L\times|e|)\times|e|}$ 将拼接后的向量映射到低维。拼接网络可以完整地保留输入顺序信息，同时，由于不需要进行矩阵乘法，拼接能够加速这个表示过程。

拼接可以认为是在 CNN 中使用宽度为 L 的卷积函数。因而从保留输入顺序信息的角度，拼接网络与池化网络处于光谱的两段。

片段差值网络（Segmental Minus，SMinus） 对片段头尾表示做差的方法在前人句法分析的部分工作[122,123] 中被证明能够作为上下文的一种表示。这种做法在与双向 LSTM 上下文表示配合使用时效果更加明显。Zhou 等人在 2017 年的文献 [86] 中也验证了这一方法在中文分词上的有效性。本章参考前人工作，并使用头尾差

值作为片段表示，即

$$s^{(\text{comp})} = (\mathbf{h}_{u,1...\frac{|\mathbf{h}|}{2}-1} - \mathbf{h}_{v,1..\frac{|\mathbf{h}|}{2}-1}) \oplus (\mathbf{h}_{v,\frac{|\mathbf{h}|}{2}...|\mathbf{h}|} - \mathbf{h}_{u,\frac{|\mathbf{h}|}{2}...|\mathbf{h}|})。 \tag{3-8}$$

公式3-8中将 $\mathbf{h}_i$ 分成两部分的做法使得差值网络能够捕捉前后两个方向的上下文表示。

3.4.2 通过片段向量表示片段

对于分割问题，片段作为一个整体时往往比其中的 n-gram 具有更强的表达能力，并且歧义更少。在前人关于 semi-CRF 的工作中，片段级的特征往往能带来性能的提升。第 3.4.1节中的片段表示只建模了输入单元（或者说输入单元的 n-gram）。因而，加入片段级的表示（即片段向量）有望进一步提升性能。

本章将片段向量视作一种与词向量类似的模块。模型将整个片段作为键值输入查找表，获得对应的片段向量。本章将片段 $s = (u, v, y)$ 的片段向量形式化地定义为：

$$s^{(\text{emb})} = \phi^{(\text{seg})}(x_{[u:v]})。 \tag{3-9}$$

其中 $\phi^{(\text{seg})}$ 是片段向量表。

在获得片段向量后，本章尝试两种使用方法：1）使用片段向量作为唯一片段标识；2）将其与第3.4.1节获得的基于输入单元组合的片段向量拼接在一起，形成片段表示。片段表示的作用可以类比一系列基于词典的切分模型。这类模型的一大特点是词典的质量影响模型的性能。由于切分的一个目标是识别片段边界，而一个片段在片段向量表中这一事实将为识别其边界的非常强的线索。所以只从训练数据中构建词典有非常大的过拟合的风险。

一个理想的查找表在尽可能包含正确切分的情况下，也应包含足够多的错误的片段，从而防止过拟合训练数据。本章提出了两种查找表的构建方法。其中一个方法是*从训练数据中构建*查找表。这种方法除了包含正确片段，也包含训练数据中的高频的错误片段。[①] 另一种方法是*从生语料中构建*查找表。这种方法使用基线模型分析大规模生语料，然后根据自动分析结果构建查找表。

第二种构建查找表的方法可以追溯到使用使用自动分析结果提到模型性能的半监督学习算法中[124,125]。这一方法的主要动机是使模型能够同时考虑正确的片段以及根据模型产生的最可能出错的片段。

词向量技术是基于神经网络自然语言处理的基石。在本章提出的第二种方法中使用词向量技术，从自动分析结果中预训练片段向量是一种潜在的有效地策略。

① 本章的‘‘错误片段’’指的是边界错误的片段。

这种预训练的片段向量即可以用作固定特征，也可以用来初始化模型。在第二种方法中，首先将自动识别出的片段视作一个新的‘‘词’’，然后使用 Word2vec 算法在这一语料上预训练片段向量。在本章实现中，本章将自动识别出的片段内的词用下划线连接，从而获得用以训练上下文相关词向量的新‘‘词’’。本章关注片段向量在识别边界时的作用，因而在学习片段向量时并没有考虑对应标签。

3.4.3 实现与模型训练细节

3.4.3.1 输入表示

为了获得输入表示，本章参考文献 [7] 并使用两类词向量 — 固定的预训练词向量 $\phi^{(\text{fix})}$ 与可调的词向量 $\phi^{(\text{tune})}$ 的拼接作为输入，即：

$$\phi(x) = \phi^{(\text{fix})}(x) \oplus \phi^{(\text{tune})}(x)。 \tag{3-10}$$

对于包含词性标签的任务 — 组块识别（chunking）和命名实体识别（NER），本章也使用词性向量 $\phi^{(\text{tag})}(\text{POS}(x))$ 作为额外的词表示并将其与公式3-10的词向量拼接形成最终词向量。

在获得了 $\phi(x)$ 后，本章使用公式3-3与公式3-4 的方法获得上下文表示。

3.4.3.2 片段表示

给定一个片段 $s = (u, v, y)$，本章将其上下文表示 $(e_u, \ldots, e_v)$ 输入到片段组合网络（公式3-5–3-8）中获得片段表示 $s^{(\text{comp})}$。同时，本章通过子串 $x_{[u:v]}$ 获得片段向量 $s^{(\text{emb})}$。然后 $s^{(\text{comp})}$ 与 $s^{(\text{emb})}$ 或拼接或独立地用作最终片段表示。除了 $s^{(\text{comp})}$ 与 $s^{(\text{emb})}$，两个片段级特征 $\phi^{(\text{len})}$ 与 $\phi^{(\text{label})}$ 也被应用于片段表示中。其中 $\phi^{(\text{len})}$ 将片段长度转化为向量；$\phi^{(\text{label})}$ 将标签 y 转化为向量。最后，本章将全部表示输入包含 ReLU 激活函数的前馈神经网中获得最后的片段表示，即

$$\begin{aligned} s^{(\text{rep})} =& s^{(\text{comp})}(e_u, \ldots, e_v) \oplus s^{(\text{emb})}(x_{[u:v]}) \oplus \phi^{(\text{len})}(u - v) \oplus \phi^{(\text{label})}(y) \\ g(\mathbf{x}, s) =& \text{ReLU}(W s^{(\text{rep})} + b). \end{aligned}$$

本章的超参设置如表3-1所示。本章在公式3-6中使用从 1 到 6 共 6 种宽度的卷积函数。[①]

本章通过优化训练数据的最大似然学习模型参数，即 $\arg\max_\theta \sum_i \log p(\mathbf{s}_i \mid \mathbf{x}_i)$。在训练所有的 LSTM 模型时，本章都是用了 recurrent dropout[107] 本章使用默认参数的 Adam 算法[104] 训练模型参数。最佳的迭代轮次有开发集性能决定。

① 对应的数量为 32, 32, 32, 16, 8, 8

表 3-1 本章超参设置
Table 3-1 The hyper-parameter settings

英文固定输入的维度 $\phi^{(\text{fix})}$	300
中文固定输入的维度 $\phi^{(\text{fix})}$	100
可调输入的维度 $\phi^{(\text{tune})}$	32
ELMo 的维度 $\phi^{(\text{ELMo})}$	1,024
词性向量的维度 $\phi^{(\text{tag})}$	12
标签的维度 $\phi^{(\text{label})}$	20
长度特征的维度 $\phi^{(\text{len})}$	4
双向 LSTM 的隐层大小	128
双向 LSTM 的层数 (公式3-3)	1
SRNN 中双向 LSTM 的层数	1
最大片段长度	10
片段向量的维度 $\phi^{(\text{seg})}$	50
batch size	32
dropout rate	0.1

3.5 基于上下文相关词向量的半-马尔科夫条件随机场

上下文相关词向量[12–16] 也被证明能够有效地提升多种自然语言处理任务的性能。本章尝试在半-马尔科夫条件随机场中使用基于语言模型的上下文相关词向量。本文已在第1.2.2中探讨了基于语言模型的上下文相关词向量的建模与参数学习。在这一章中，本章按照公式1-6 将上下文相关词向量用作 ‘‘特征’’，即固定其模型，只调整公式1-6中的 γ 与 s_k。

本章从两个角度探讨上下文相关词向量在 semi-CRF 中的应用。其中一种是将其作为上下文表示的替代，即将公式3-3替换为

$$(\mathbf{h}_1, \ldots, \mathbf{h}_n) = \text{ELMo}\,(x_1, \ldots, x_n)\,。 \tag{3-11}$$

另一种是将其作为输入词向量的替代，即将公式3-3替换为

$$(\mathbf{h}_1, \ldots, \mathbf{h}_n) = \text{BiLSTM}(\text{ELMo}\,(x_1, \ldots, x_n))。 \tag{3-12}$$

后者可以认为是在通用上下文表示的基础上融入任务相关的上下文表示。

3.6 实验

3.6.1 任务

本章在三个典型的 NLP 切分任务 — 组块识别、命名实体识别和中文分词上进行实验。本章表示数据的统计信息如表3-2所示。

组块识别 对于组块识别实验，本章使用 CoNLL00 数据集[105] 进行实验。CoNLL00 采用华尔街时报（Wall Street Journal data，简称 WSJ，文献 [65]）数据后处理获得。

表 3-2 各数据集的标注数据统计
Table 3-2 The statistics of labeled data

		CoNLL00	CoNLL03	CTB6	PKU	MSR
句子数量	训练集	8,936	14,987	23,416	17,149	78,232
	开发集	1,844	3,466	2,077	1,905	8,692
	测试集	2,012	3,684	2,796	1,944	3,985
词数量	训练集	211.7K	204.6K	1,055.5K	1,662.6K	3,633.3K
	开发集	44.4K	51.6K	100.3K	163.9K	417.1K
	测试集	47.4K	46.7K	134.1K	172.7K	184.4K

其中 15-18 节用作训练数据，20 节用作测试数据。由于原始数据缺少开发集，本章利用官方提供的短句结构句法到组块识别的工具 `chunklink.pl`① 讲 22 节转化为组块分析的开发集。

命名实体识别 对于命名实体识别实验，本章使用 CoNLL03 数据集[64] 进行实验。该数据集被广泛用于评估 NER 模型的性能。本章采用 F-score 用作评估指标。相应指标通过 CoNLL03 share task 的脚本获得。②

中文分词 对于中文分词，本章参考前人工作并使用三个简体中文数据集。这三个数据集包括第二次 SIGHAN 中文分词评测（the second SIGHAN bakeoff）中的 PKU 和 MSR 数据集，以及 Chinese Treebank 6.0（CTB6）。对于 PKU 和 MSR 数据集，本章参考文献 [126]，并采用训练数据的最后 10%作为开发数据。对于 CTB6 数据，本章使用标准方式划分训练、开发和测试数据。在预处理中，本章将 PKU 数据中的所有双字节数字和字母转换为单字节。与命名实体识别相同，本章采用 F-score 评估中文分词性能。③

词向量 本章使用文献 [3] 中开源的 840B GloVe 词向量作为固定的英文词向量。本章使用文献 [58] 开源的考虑相对距离的 skip-gram 工具④在 Chinese Gigawords Version 5 生语料上获得中文字向量。

生语料 本章在大规模生语料上按照第3.4.2节提出的方法获得片段向量。对于英文实验，本章使用 RCV1 作为未标注数据。对于中文实验，本章使用 Chinese Gigawords Version 5 作为未标注数据。表3-3显示了生语料的统计信息。

① https://github.com/esrel/DP/blob/master/bin/chunklink.pl
② 本章使用 `conlleval` 脚本。
③ 本章使用第二次 SIGHAN 中文分词评测提供的 `score` 脚本。
④ `https://github.com/wlin12/wang2vec`

表 3-3 生语料的统计信息
Table 3-3 The statistics of unlabeled data

	RCV1	Gigawords-v5
句子数量	8.18M	4.82M
词数量	131.22M	473.73M

文献 [84] 指出神经网络的训练过程是非确定的，并且严重依赖于初始化。为了控制初始化对于模型性能的影响，本章对于所有参数均进行了 5 组不同随机种子的实验，并报告了实验的均值和方差。

3.6.2 基线系统

本章将本章提出的模型与五个基线系统进行了对比，这些系统包括：

- *Sparse-CRF*：使用人工定义特征的 CRF 模型；
- *NN-Labeler*：使用神经网络对输入的上下文进行建模并对每个输入独立分类的模型；
- *NN-CRF*：使用神经网络对输入的上下文进行建模并对整个序列采用 CRF 建模的模型；
- *SRNN*：Kong 等人在文献 [119] 中提出的使用 RNN 建模片段的基于神经网络的 semi-CRF 模型。为了公平比较，本章在采用相同的输入表示、上下文表示的前提下重新了他们的模型。
- *grRecNN*：Zhuo 等人在文献 [120] 中提出的使用递归神经网建模片段的基于神经网络的 semi-CRF 模型。与 SRNN 类似，本章也在相同的设置下重现了他们的模型。

对于使用 CRF 和分类的基线系统，本章使用 BIESO 标签体系建模分割问题。[①] 对于 *Sparse-CRF*，本章使用文献 [127] 中的基线特征模板建模命名实体识别任务，并使用文献 [128] 提出的特征模板建模中文分词任务。*NN-Labeler* 和 *NN-CRF* 都采用与本章提出的 Semi-CRF 模型相同的输入单位向量，但在解码上有所不同，并且没有显示地对片段级信息建模。

3.6.3 组合函数的对比

根据表3-4，SRNN 和 SConcat 的结果比较接近。并且性能优于 SCNN。尽管 CNN 可以对任何长度的输入序列建模，但其与精确位置的不变性可能是表示段的缺陷。实验结果证实了并且显示了正确处理输入位置的重要性。考虑到 SCNN 的性能相对较差，我们仅在以下实验中研究 SRNN 和 SConcat。

① O 标签代表无标签片段。由于中文分词不涉及给片段打标签，故在实验中不采用 O 标签。

表 3-4 在使用不同组合函数情况下，组块识别、命名实体识别以及中文分词的对比实验。本章报告了 5 次实验的均值，± 后的数字表示 5 次实验结果的标准差。*spd.* 代表相对 NN-Labeler 所需的解码时间的倍数。

Table 3-4 The chunking, NER, and CWS results of the baseline models and our neural semi-CRF models with different input composition functions. The number after ± shows the standard variance. *spd.* represents the inference speed and is evaluated by running time against that of NN-Labeler.

模型	CoNLL00	CoNLL03	CTB6	PKU	MSR	Ave.	spd.
NN-Labeler	93.31 (±0.14)	88.46 (±0.18)	93.12 (±0.08)	92.87 (±0.10)	95.19 (±0.45)	92.66	1.00
NN-CRF	93.84 (±0.09)	88.83 (±0.18)	93.64 (±0.09)	93.57 (±0.04)	95.47 (±0.07)	93.15	1.66
Sparse-CRF	93.29	83.43	**95.08**	**95.06**	**96.54**	92.66	
raw embeddings (Eq. 3-4)							
SRNN	93.31 (±0.14)	83.37 (±0.29)	94.35 (±0.14)	**94.26** (±0.07)	**96.51** (±0.07)	**92.36**	14.01
grRecNN	**93.07** (±0.12)	81.64 (±0.42)	**94.47** (±0.07)	94.24 (±0.13)	96.10 (±0.08)	91.90	7.00
SAve	90.15 (±0.24)	81.75 (±0.50)	91.38 (±0.19)	90.61 (±0.08)	92.26 (±0.20)	89.23	2.24
SCNN	92.93 (±0.02)	**86.07** (±0.85)	93.90 (±0.07)	93.49 (±0.09)	95.69 (±0.03)	92.33	12.27
SConcat	92.46 (±0.04)	82.41 (±0.77)	94.22 (±0.14)	94.12 (±0.11)	95.78 (±0.05)	91.80	1.70
SMinus	88.76 (±0.19)	81.67 (±0.32)	91.18 (±0.13)	90.64 (±0.14)	90.89 (±0.23)	88.63	1.33
BiLSTM (Eq. 3-3)							
SRNN	94.25 (±0.06)	88.88 (±0.61)	94.14 (±0.06)	93.91 (±0.11)	95.98 (±0.10)	93.49	16.58
grRecNN	**94.43** (±0.06)	89.07 (±0.36)	94.43 (±0.09)	94.18 (±0.09)	95.96 (±0.04)	**93.66**	16.78
SAve	94.11 (±0.20)	89.02 (±0.12)	93.74 (±0.13)	93.53 (±0.08)	95.39 (±0.10)	93.20	5.19
SCNN	94.16 (±0.26)	**89.10** (±0.61)	94.17 (±0.06)	94.08 (±0.10)	95.74 (±0.08)	93.45	11.90
SConcat	94.31 (±0.07)	88.69 (±0.65)	**94.62** (±0.05)	**94.48** (±0.07)	**96.11** (±0.07)	93.64	4.68
SMinus	93.87 (±0.17)	88.21 (±0.27)	93.62 (±0.09)	93.55 (±0.10)	95.05 (±0.09)	92.86	3.50

对推理速度的进一步比较表明，SConcat 运行速度比 SRNN 快 1.7 倍，但比基于神经网络的分类模型以及基于神经网络的 CRF 模型慢，这是由于时间复杂度的内在差异造成的。

本章首先研究了不同组合函数（第3.4.1节）在表示片段时的性能。本章研究的内容包括，不同的上下文表示以及不同组合函数的对比。组块识别、命名实体识别以及中文分词的实验结果如表3-4所示。表3-4的第一个组结果显示了序列标注与 CRF 模型的结果。第二组显示不使用上下文表示（公式3-4）时的模型的性能。第三组显示使用双向 LSTM 进行上下文表示（公式3-3）是的模型性能。在表3-4的各个比较中，本章提出的 SConcat 在使用双向 LSTM 进行上下文表示时取得了与最优系统（grRecNN）几乎一致的性能。在比较的几个任务中，grRecNN 在组块识别中表现最好，SCNN 在命名实体识别中表现最好。而得益于对 ngram 特征的表示，Sparse-CRF 在分词中表现最好。[①]

通过对比本章的第三组中的 semi-CRF 与 NN-CRF，本章发现，在除了 SMinus

① 文献 [129] 提出使用字的 bigram 作为输入能取得更好的中文分词效果。但是出于模型的统一性，本章仍将字作为输入。

之外的设置中，semi-CRF 模型的性能都好于 linear-chain CRF。semi-CRF 与 linear-chain CRF 的最大平均分差异是 0.55。这一结果显示了在分割问题中建模片段的重要性。

通过对比第二组和第三组结果，本章发现在 semi-CRF 中建模上下文的重要性。在对比的任务中，建模上下文在 CoNLL00 与 CoNLL03 上给模型性能带来更大的提升。[①] 相比之下，中文分词数据集的提升相对较小[②] 这一观察表明组块识别与命名实体识别对于上下文信息更敏感。虽然提升大小随任务不同而不同，本章认为在 semi-CRF 中建模上下文是必要的。

通过对比第三组中不同的片段组合函数，本章提出的 SConcat 的性能只弱于 grRecNN 平均分 0.02。更值得关注的是，本章关注的建模顺序的模型（SRNN、grRecNN 以及 SConcat）的效果好于不关注顺序的模型（SAve 与 SMinus）。这一观察表明通过组合输入表示片段的过程中建模顺序的重要性。

本章进一步比较了不同模型的解码速度。其中 SConcat 的速度三倍以上快于 SRNN 与 grRecNN。但相对 NN-Labeler 和 NN-CRF 仍有差距。本章认为这一差距是算法内在复杂度导致的。

考虑到 SConcat 的准确率与性能的优势，本章在后续实验中都使用 SConcat 作为本章的输入组合函数。

3.6.4 片段向量的对比

在这一节中，本章研究使用片段向量作为唯一片段表示的问题。

根据第3.4.2节，片段表示查找表的构建对表示的性能发挥了重要作用。本章首先研究了*从训练数据中构建*查找表的效果并经验性地验证了前文的讨论。本章在 CoNLL03 与 CTB6 上进行了实验并绘制了片段负例的个数与模型性能的关系。这一关系见图3-3。根据图3-3，模型性能随着负例的增加而增加。这一观察反映了负例个数与最终结果的关系。然而，图3-3 的结果较差，一定程度上表明*从训练数据中构建*查找表的方法并不是最优方法。有效地构建查找表的方法尚需探索。

接下来，本章研究了*从生语料中构建*查找表。根据前文，这种方法首先使用基线模型自动分析大规模生文本，并从分析结果中构建片段查找表。因而基线模型在这类方法中扮演重要的作用。本章尝试使用两种基线模型 — 基于神经网络的 semi-CRF（记为 Baseline）以及 Sparse-CRF（记为 CRF）。与前文类似，本章在这里也在 CoNLL03 和 CTB6 上进行实验。前文也谈及预训练的片段向量的问题。本章在这处实验中也分析了预训练的片段向量对模型性能的影响。这些分析包括 —

① 对应的提升为 1.36 和 3.03。

② 对应的提升为 0.15、0.22 以及-0.40。

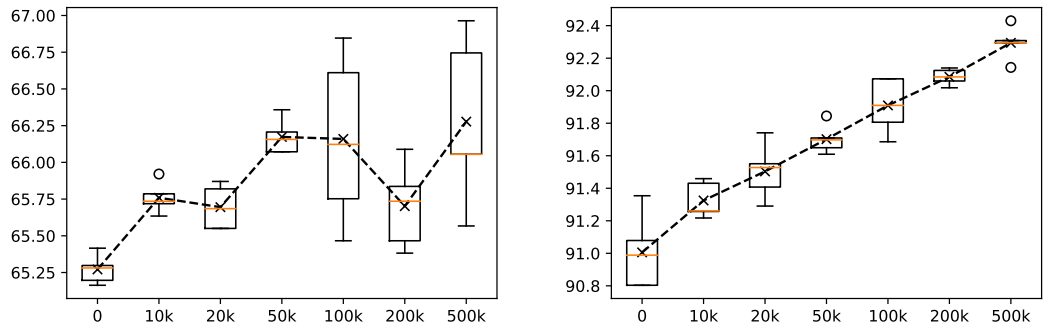

图 3-3 片段查找表中负例的个数与模型性能的关系。左图显示的是 CoNLL03 的结果，右图显示的是 CTB6 的结果。

Fig. 3-3 The relation between number of negative segments and the model's performances. The left figure plots that for CoNLL03 and the right figure plots that for CTB6.

表 3-5 不同查找表构建方法的以及片段向量用法的对比。

Table 3-5 The effect of different segment embeddings, including use them as fixed one, update their weights with and without initialization.

模型	CoNLL03	CTB6
500K update	51.37 (±0.38)	92.13 (±0.11)
Baseline 预训练片段向量		
固定	**74.81** (±0.41)	91.86 (±0.10)
初始化并更新	73.62 (±0.51)	93.65 (±0.13)
不初始化但更新	63.54 (±1.54)	92.96 (±0.08)
CRF 预训练片段向量		
固定	73.38 (±0.14)	92.05 (±0.25)
初始化并更新	73.03 (±0.41)	**93.68** (±0.08)
不初始化但更新	62.34 (±0.74)	93.13 (±0.15)

1）固定预训练片段向量；2）将预训练片段向量作为初始化；3）只使用片段向量查找表。其中第三种做法可以与从训练数据中构建查找表的方法进行公平对比。这一实验结果如表3-5所示。即使只使用片段向量查找表，本章的从生语料中构建查找表的方法仍取得了优于从训练数据中构造查找表的性能。而使用预训练的片段向量能进一步提升性能。这一结果显示了本章提出的从生语料中构建查找表的有效性。

表3-5也显示了是否需要更新片段向量的对比。然而，根据表中所示结果，是否更新片段向量与任务相关。CoNLL03 上固定向量效果更好，而 CTB6 更新向量效果更好。表3-5也显示了不同基线模型的比较。结果也没有明显趋势，同时不同的基线性能相差不多。这一结果表明本章提出的从生语料中构建查找表的方法对于产生自动分析数据的方法并不敏感。

本章使用 t-SNE[130] 对 NER 的片段向量进行了可视化。其结果如图3-4所示。

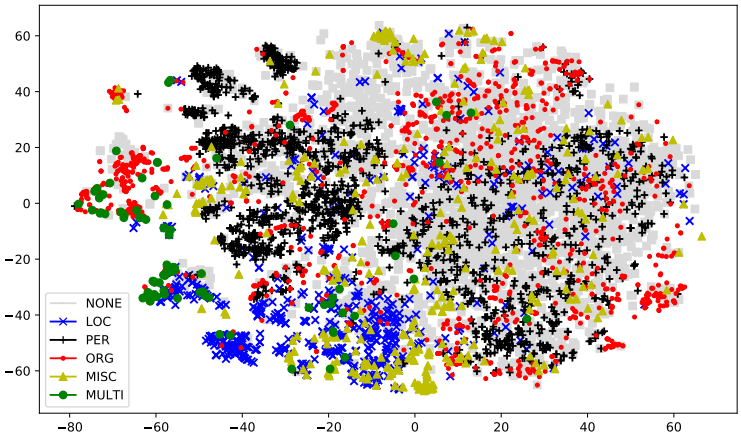

图 3-4 CoNLL03 片段向量的可视化结果。片段的颜色代表对应的标签，“NONE’’ 代表错误向量，“MULTI’’ 代表多标签片段。

Fig. 3-4 Visualization on the segment embeddings for CoNLL03. “NONE” denotes the incorrect segment and “MULTI” denotes the segment that is associated with multiple labels in CoNLL03.

图3-4显示片段向量能够从一定程度上刻画片段的标签信息。这一观察部分解释了固定片段向量或将其用作初始化能够带来提升的原因。然而，图3-4显示片段表示并不能将错误片段（标记为 NONE 的片段）与正确片段进行有效区分。本章认为这一现象的原因是错误片段具有稀疏性，因而无法学得合理的片段向量。更重要的是，Word2vec 算法的所基于的分布式假设无法保证将错误片段与正确片段进行合理区分。

3.6.5 片段表示组合

接下来，本章研究了将输入组合获得的表示与片段向量进行组合从而获得片段表示的效果。本章实验结果如表3-6所示。根据表3-6，使用片段向量能够给模型带来显著的性能提升。相对 NN-CRF 基线模型，在多个任务上的平均提升为 1.46，平均错误率降低为 24.56%。相对于只使用输入组合方式进行片段表示的模型，平均提升与平均错误率降低分别为 0.97 与 17.61%。这一比较现实了建模片段整体的重要性。

除了性能提升，本章从表3-6也观察到片段向量带来的提升是任务相关的。片段向量未给组块识别带来显著性能提升，但给命名实体识别和中文分词带来了多于一个点的提升。本章将这一观察归因为组块识别中需要识别的片段的多样性更高。这使得片段向量无法有效地在查找表中召回足够多的片段。本章将片段多样性定义为如下算式：

$$\text{Diversity} = \frac{\text{\# of unique segments}}{\text{\# of segments}}。 \tag{3-13}$$

表3-6的倒数第三行显示了不同数据集的片段多样性。根据表3-6，CoNLL00 数据

表 3-6 使用组合输入与片段向量两种方式结合的实验结果。第一行的错误率降低是相对 NN-CRF 而言的，第二次错误率降低是相对不使用片段向量的模型而言的。

Table 3-6 The results of using both the composition function and segment embeddings. The first error reduction is calculated between the model with segment embeddngs and NN-CRF. The second error reduction is calculated between the model with and without segment embeddngs.

模型	CoNLL00	CoNLL03	CTB6	PKU	MSR
开发集结果					
NN-CRF	95.26	92.92	94.26	94.55	95.42
BiLSTM+SConcat	95.85	93.13	95.30	95.67	96.00
+ 固定的片段向量 (Baseline)	95.86	93.52	96.05	96.83	97.07
+ 固定的片段向量 (CRF)	**95.86**	93.52	96.12	96.98	97.46
+ 更新的片段向量 (Baseline)	95.74	93.77	**96.22**	**97.03**	97.45
+ 更新的片段向量 (CRF)	95.77	**93.79**	96.15	97.02	**97.70**
测试集结果					
NN-CRF	93.84	88.83	93.64	93.96	95.47
BiLSTM+SConcat	94.31	88.69	94.62	94.48	96.11
+ 由开发集结果决定的片段向量	**94.39** (±0.09)	**89.87** (±0.41)	**95.63** (±0.05)	**95.67** (±0.11)	**97.52** (±0.07)
Diversity	27.44	10.77	6.08	4.75	3.64
错误率降低 [1]	8.88%	9.27%	31.17%	28.27%	45.21%
错误率降低 [2]	1.30%	10.40%	18.61%	21.50%	36.22%

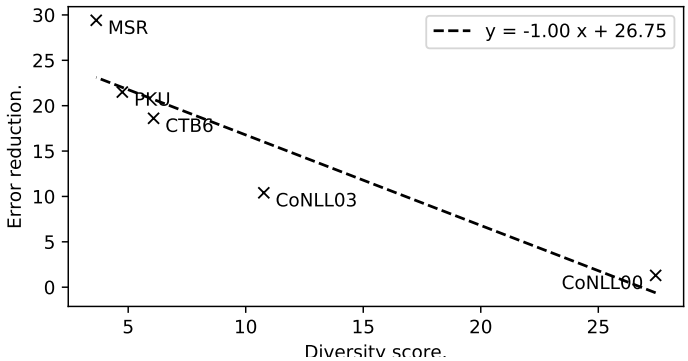

图 3-5 片段多样性与相对误差降低之间的关系。

Fig. 3-5 The relation between diversity and error reduction.

集的多样性最高。本章也绘制了多样性与错误率降低的关系。其关系如图3-5所示。两者的皮尔逊相关系数为 −0.84，这一结果进一步验证了前文的假设。

由于切分涉及识别边界和并给片段打标签，本章将错误分为两类：边界错误（即边界识别错误）和标签错误（即边界识别正确但标签错误）。本章研究了使用片段向量的模型在这两个错误中的效果。表3-7显示相比标签错误，使用片段向量的模型纠正更多的片段错误。这一观察结果表明，合理地获得片段向量能够以进

表 3-7 使用片段向量带来的错误分布的变化。这一实验结果是在 CoNLL03 上获得的。
Table 3-7 The distribution of two types of errors for our model with and without segment embeddings. The numbers are obtained on the results of multiple runs on CoNLL03.

模型	边界错误	标签错误
BiLSTM+SConcat	354.00	285.60
+ 片段表示	310.75	269.15
Error reduction	12.25%	5.76%

一步提高性能。

然而，表3-7显示：使用怎样的基线系统获得片段表示以及是否更新片段向量的影响并不明显。固定片段向量对于组块识别更有效；而更新片段向量对于其他任务更有效。只有 MSR 数据集上的实验显示使用 sparse-CRF 获得自动分析的片段的有效性。

至此，本章经验性地分析了对于基于神经网络的 semi-CRF 的重要部分。根据实验结果，本章得出以下结论：

- 基于神经网络的 semi-CRF 模型在大部分切分任务中性能好于 linear-chain CRF（参考表3-4与3-8结果）；
- 即使输入表示中编码了通用领域的上下文信息，对于特定任务的上下文进行编码仍是重要的（参考表3-4与3-8结果）；
- 使用片段向量的方法能够有效帮助片段多样性不过大的任务（参考表3-5、3-6以及3-8结果）。

3.6.6 基于上下文相关词向量的半-马尔科夫条件随机场

通过语言模型学习的上下文相关词向量（Embeddings from Language Model，简称 ELMo，[13]）为多项自然语言处理任务带来了提升。本章尝试使用上下文相关词向量作为输入。[①] 对于英文数据，本章使用文献 [13] 中开源的使用 *1 Billion Word Benchmark* 数据训练的上下文相关词向量作为输入。[②] 对于中文字符，本章使用第2章的 ELMo 工具在 Chinese Gigawords V5 上训练字级别 ELMo。[③] 本章实验中使用的 ELMo 的维度为 1,024。这种表示严重过参数（over-parameterized）因而很容易过拟合到训练数据。所以本章在这处实验中将 dropout rate 调整为 0.25。

表3-8显示了这部分实验结果。本章观察到表3-8与表3-6的趋势相同，即本章 semi-CRF 模型在 5 个实验数据集的 3 个中取得了优于 NN-CRF 的效果。使用片段表示进一步给 4 个数据集带来性能的提升。片段多样性与相对错误率降低的关系

① 具体来讲，本章将 ELMo 作为公式3-10中 $\phi^{(\text{fix})} \oplus \phi^{(\text{tune})}$ 的替代。
② https://allennlp.org/elmo
③ https://github.com/HIT-SCIR/ELMoForManyLangs/

表 3-8 使用 ELMo 作为输入的实验结果。最后两行的错误率降低与表3-6含义相同。

Table 3-8 The test results with model using ELMo as input representation. The first and second error reduction have the same meanings with those of Table 3-6.

model	CoNLL00	CoNLL03	CTB6	PKU	MSR
NN-CRF	96.48 (±0.10)	91.72 (±0.28)	95.91 (±0.05)	95.58 (±0.03)	96.74 (±0.02)
SConcat	96.07 (±0.12)	89.31 (±0.35)	95.57 (±0.07)	94.78 (±0.11)	95.30 (±0.10)
BiLSTM+SConcat	**96.60** (±0.06)	91.84 (±0.07)	96.11 (±0.12)	95.39 (±0.05)	96.75 (±0.05)
+ 由表3-6开发集结果决定的片段表示					
	96.58 (±0.09)	**92.33** (±0.05)	**96.46** (±0.13)	**95.97** (±0.08)	**97.85** (±0.06)
Diversity	27.44	10.77	6.08	4.75	3.64
错误率降低 [1]	2.73	7.36	13.23	8.94	33.92
错误率降低 [2]	-0.67	5.91	8.81	13.12	33.81

也符合图3-5的趋势。使用片段表示并未给 CoNLL00 数据集带来显著性能提升。本章认为这一现象是由 CoNLL00 的片段多样性较高造成的。

本章也在表3-8中比较了直接将 ELMo 用于片段组合的模型（即不在模型中使用 BiLSTM 表示上下文），其结果显示在 “SConcat’’ 一列。相比表3-4，使用 ELMo 能够给性能带来提升。本章认为这一提升源于 ELMo 在通用领域建模了上下文信息。然而，这一结果相较使用任务相关的上下文表示的模型仍有差距。这一观察表明进行任务相关的上下文表示的重要性。

根据表3-8，相对 NN-CRF，使用片段向量能够带来 13.16 的平均错误率降低。而相对不使用片段向量的模型，平均错误率降低是 12.00。对应的平均性能绝对值提升为 0.55 和 0.49。表3-8中显示的错误率降低小于表3-6。这一结果表明，输入组合函数与片段向量的作用不是完全正交的。在加强输入表示的情况下，片段向量的作用被相对削弱。然而，片段向量仍给本章模型带来了性能的提升。这一观察显示了本章提出的片段向量的有效性。

3.6.7 与当期最优模型对比

本章也将本章的基于神经网络的 semi-CRF 模型与当前最优模型（state-of-the-art，简称 SOTA）进行了对比。表3-9显示了组块识别的对比结果。本章基于神经网络的 semi-CRF 模型的表现优于除多任务学习[82] 以及使用丰富字级别特征[109] 之外的大部分模型。由于文献 [82] 中的多任务学习存在一个任务的测试集是另一个任务的训练集的情况，其结果并不能直接比较。① 通过与文献 [109] 的对比，本章认为通过加强输入表示，本章的模型有望取得更好的效果。

表3-10显示了命名实体识别的比较结果。这一结果与表3-9结果的趋势一致。即

① 在文献 [82] 的多任务学习中，依存句法分析使用 WSJ 2 到 21 节作为训练数据。而 CoNLL00 使用 WSJ 20 节做为测试数据，这使得模型学习过程中观察到测试数据。

表 3-9 组块识别的对比结果。♣ 代表使用多任务学习的系统；♡ 代表使用半监督学习的系统；∗ 的系统与本章系统无法直接比较；♢ 代表结果是多次实验的平均值。

Table 3-9 Comparison with the state-of-the-art chunking systems. ♣ marks the system that uses multi-task learning. ♡ marks the system that uses semi-supervised learning. ∗ marks the result which is not directly comparable due to data split difference. ♢ marks the result which is obtained from the average of multiple runs.

模型	CoNLL00
CVT+Multi-task (Large)$^{\heartsuit\clubsuit\diamond}$ [82]	**97.0**
Flair embeddings$^{\heartsuit\diamond}$ [109]	96.72
TagLM$^{*\heartsuit\diamond}$ [83]	96.37
LM-LSTM-CRF$^{*\heartsuit\clubsuit\diamond}$ [131]	95.96
JMT$^{\clubsuit}$ [132]	95.77
Low supervision [133]	95.57
Suzuki and Isozaki (2008)$^{\heartsuit}$ [134]	95.15
grRecNN (reported)$^{\heartsuit}$ [120]	95.10
NCRF++ [135]	95.06
ours$^{\heartsuit\diamond}$	96.58

使用丰富输入表示[16,109] 以及多任务学习[82] 可以给命名实体识别带来更好的性能。表3-10显示，本章系统只比当前最优的应用 BERT[16] 的命名实体识别系统低 0.47。本章认为这一差距主要是由于学习上下文相关词向量时数据量的差异（BERT 使用 330 万，ELMo 使用 80 王），以及 BERT 的命名实体识别使用词片段以及篇章级表示。然而，值得注意的是本章模型比文献 [13, 83] 中使用 ELMo 的模型效果好。这显示本章使用片段向量的有效性。

本章在表3-11中将本章系统与其他最先进的中文分词模型进行了比较。表3-11显示了使用字符 bigram 作为输入对于前人工作的重要性。本章在只使用 unigram 的前提下取得了与前人最优系统相近的性能。在 CTB6 上，本章模型只低于文献 [93] 提出的当前最优的精细调参 biLSTM-CRF 模型低 0.24。在 PKU 上，本章与前人最优系统的差距是 0.33，在 MSR 上，这一差距是 0.25。需要指出的是，表3-11 中的所有前人工作均为单一结果。而本章报告了多次运行的平均值，因而报告的结果更稳定可靠。

本章提出的模型在三个中文分词任务上相对前人最优系统的平均差距为 0.29。可以认为本章提出的模型达到了与前人最优系统相近的性能。这一结果再次显示了使用 semi-CRF 建模分割问题，并合理表示片段的有效性。

3.7 相关工作

Semi-CRF 已经在诸如信息抽取[118]、命名实体识别[115]、意见挖掘[116]、流利检测[144] 以及中文分词[117,143] 等多项 NLP 任务中取得了成功。然而，这些工作大

表 3-10 命名实体识别的对比结果。♣、♡ 以及 ♢ 与表3-9 含义相同。∗ 的系统由于数据集划分与本章系统无法直接比较。

Table 3-10 Comparison with the state-of-the-art NER systems. ♣, ♡, and ♢ have the same meanings with those in Table 3-9. ∗ marks the result which is not directly comparable due to using train and development splits for training.

模型	CoNLL03
Flair embeddings*♡♢ [109]	93.09
BERT Large♡ [16]	**92.8**
CVT+Multi-task (Large)♣♡♢ [82]	92.6
BiLSTM-CRF+ELMo♡♢ [13]	92.22
TagLM*♡♢ [83]	91.93
HSCRF♡♢ [136]	91.38
NCRF++ [135]	91.35
LM-LSTM-CRF♡♣♢ [131]	91.24
BiLSTM-CRF-CNN [23]	91.21
LSTM-CRF [6]	90.94
grRecNN (reported)♡ [120]	90.87
ours♡♢	92.33

多采用离散特征作为输入。semi-CRF 与神经网络的结合的研究相对较少。文本基于文献 [119, 120, 136] 中的基于神经网络的 semi-CRF 工作，并进行了若干有意义的拓展。除此之外，本章详细地研究了基于神经网络的 semi-CRF 中的重要组成部分。

序列标注是一种常用的分割任务的转化方式。前人研究表明，通过丰富输入表示，即使使用简单的结构模型，序列标注模型可以也取得很好的分割准确率[13,16,83,109,131]。本章在使用 ELMo 作为输入的情况下也观察到了性能的提升，这证明即使在很强的表示模型中，合理地建模结构仍然对模型有帮助。

使用自动分析的数据帮助中文分词的已经在文献 [125] 中得到研究。但他们的工作基于特征工程，同时只从自动分析的数据中求算统计量作为特征，而不关注自动分析的结果是什么。得益于近期词向量[2] 的相关工作，本章提出的方法可以显示地完整地表示自动分析的结果。

本章将本章方法在三个自然语言分割任务上进行了实验。其中包括组块识别、命名实体识别以及中文分词。然而，很多自然语言处理任务也可以建模为分割问题，比如流畅检测[144]、信息抽取以及对话系统中的语义理解[145]。在这些任务中使用本章方法也是一个值得探索的。

3.8 本章小结

本章全面地研究了在 semi-CRF 模型中使用神经网络表示片段的问题。本章

表 3-11 中文分词的对比结果。† 代表使用字 unigram 做输入的系统；‡ 代表使用 bigram 做输入的系统；♡ 以及 ♢ 与表3-9 含义相同。∗ 的系统由于预处理的不同与本章系统无法直接比较。
Table 3-11 Comparison with the state-of-the-art CWS systems. † marks the system that uses unigram character as input. ‡ marks the system that uses bigram character as input. ♡ and ♢ have the same meanings with those in Table 3-9. ∗ marks the result that is not directly comparable due to prerocessing on digits and English letters according to[137].

模型	CTB6	PKU	MSR
BiLSTM-CRF+hyper-param. search†‡ [93]	**96.7**	96.1	**98.1**
Transition-based+emb. tuning†‡ [138]	96.2	**96.3**	97.5
Greedy Search+word context† [86]	96.2	96.0	97.8
BiLSTM-CRF+adv. loss†‡ [139]		94.3	96.0
Greedy Search+Span repr.† [140]	-	95.8	97.1
Greedy Search† [85]	-	95.7	96.4
Gated Recursive NN∗†‡ [141]	95.8	96.4	97.6
LSTM∗†‡ [142]	94.9	95.7	96.4
Max-margin Tensor NN†‡ [126]	-	95.2	97.2
CNN† [21]	-	92.4	93.3
SRNN (reported)† [119]	-	90.6	90.7
Sparse semi-CRF[143]	-	95.2	97.3
Sparse transition-based[20]	-	95.1	97.2
Auto-segmented Features♡ [125]	95.7	-	-
ours†♡♢	96.46	95.97	97.85

提出了使用拼接网络将输入单元组合为片段表示。这种方法取得了与 SRNN 和 grRecNN 相近的准确率与更快的速度。本章同时提出将片段整体进行表示并将得到的片段向量作为一种额外的表示加入片段表示中。实验结果表明这一技术能够显著提高模型性能，特别是对片段多样性较低的数据。组块识别、命名实体识别以及中文分词实验证明了本章提出的模型的有效性。同时，本章也对基于神经网络的 semi-CRF 的各部分进行了全面的分析。分析显示建模任务相关的上下文对 semi-CRF 的性能至关重要。

在此基础上，本章将上下文相关词向量与 semi-CRF 进行结合，以研究其针对片段的组合能力。本章研究表明上下文相关词向量可以通过简单组合进行片段表示并获得模型性能提升。在上下文相关词向量的基础上进一步进行任务相关的上下文表示可以带来更大的提升。通过这一方法，本章模型取得了与前人最优结果相当的性能。

第 4 章 基于上下文相关词向量的句法分析

4.1 引言

从生文本中有效地学习一直是自然语言处理的目标。上下文相关词向量[12–14,16,109] 是一种有效的建模生文本的方法。它使用（广义）语言模型从生文本中学习上下文信息，并被证明是可以有效地提高包括问答[13,16]、语言角色标注[13]、共指消解[94]、文本蕴含[13,16]、情感分析[13]、短语结构句法[95,96] 以及命名实体识别[16,83] 在内的多种任务的准确率[77]。

尽管上下文相关词向量帮助一部分语义任务取得了性能的提升，但其对句法任务的作用尚不为人知。为了分析其能否带来性能提升以及性能提升的来源，理想的测试平台应包含相同标注规范下的多个句法分析数据集。比较上下文相关词向量对不同数据集的影响可以揭示性能提升与数据集内在特征之间的关系，从而更好地理解上下文相关词向量。本章选择多国语通用依存句法分析（Universal Dependency，文献 [106]）作为测试平台。选择通用依存句法分析的原因是它提供了一种跨语言句法标注规范以及 60 多种语言的超过 100 个依存句法树库。回答上下文相关词向量能否提高通用依存句法分析性能并探索这些性能提升的原因可以有助于更好地理解上下文相关词向量。

本章研究了基于语言模型的上下文相关词向量— ELMo[13] 对通用依存句法分析及其词性标注的影响。本章基于 Dozat 与 Manning 在 2016 年的文献 [8] 中提出的当前性能最优的深度双仿射依存句法分析器（Deep biaffine parser）构建本章的词性标注器与句法分析器。同时，本章使用 ELMo 作为额外的词向量来改进文献 [8] 的算法。CoNLL 2018 国际通用依存句法分析评测（CoNLL 2018 shared tasks — Multilingual Parsing from Raw Text to Universal Dependencies，文献 [146]）的 57 个树库的实验结果表明：使用 ELMo 可以给当前最优的基线系统带来稳定的提升。对于词性标注任务，ELMo 给 57 个树库带来平均 0.91 的提升；对于句法分析任务，ELMo 带来的提升是 1.11。

在获得了性能的提升后，本章进一步分析其背后的原因。通过研究不同的树库的五种特征及其与性能提升的相关性，本章发现 ELMo 主要通过改进未登录词（Out-of-vocabulary Word，简称 OOV）的表示能力来改进句法分析。这种表示能力的改进主要源于 ELMo 中基于字级别 CNN 词表示模型。对于向量的可视化进一步表明 ELMo 通过在词向量空间中将相同词性的的未登录词聚类在一起来学习更好

的未登录词抽象。

基于这些分析，本章进一步将包括 ELMo 以及本章第2章提出的基于局部信息的上下文相关词向量（SelfAttnLBL）在内的上下文相关词向量应用在资源稀缺语言的依存句法分析中。通过模拟实验，上下文相关词向量可以在少量标注数据（100 句或更少）的情况下显著提高模型性能。这一实验显示上下文相关词向量是解决资源稀缺问题的一种很有前景的技术。

本章的主要贡献如下：

• 本章在大规模数据上验证了 ELMo 对于通用依存句法分析的效果。实验结果表明，ELMo 能够为当前最优的系统带来了稳定的提升（§4.5）。

• 本章对上述结果进行了详细的分析（§4.6），并证明 ELMo 带来的性能提升主要源于更好的未登录词的抽象能力（§4.6.3），而 ELMo 的基于字的词表示模型对这种抽象能力贡献很大（§4.6.4）

• 基于上述分析，本章在资源稀缺的模拟数据中进行了实验（§4.6.5）。实验证明少量标注数据与上下文相关词向量配合可以获得很好的句法分析性能。

4.2 背景知识

依存句法分析 依存句法分析是一个基础 NLP 问题。依存句法分析的目标是识别句子中的词及其修饰词之间的关系。近年来，随着研究的深入，依存句法的形式化定义与数据集发生了很大的变化。这些变化包括从针对英语金融新闻的宾州树库数据集[65] 到适用多国语并且包含近百个树库的通用依存句法项目（Universal dependency project，文献 [106]）。通用依存句法项目对不同语言、不同风格的数据定义了一套相同的词性和依存句法关系（参考图4.2）。这一变化为依存句法分析算法提出新的挑战。

在过去的几十年中，学术界提出了多种数据驱动的依存句法分析算法。其中包括 1）基于图的算法[25] 与 2）基于转移的算法[7,26,147]。基于图的算法通过定义在子树结构上的打分函数来确定最大生成树。基于转移的算法使用概率有限状态机对依存句法树的生成过程进行建模。近年来，神经网络在 NLP 问题中取得巨大成功，这一成功也影响了依存句法分析算法。得益于神经网络强大的表示能力，目前最先进的依存句法分析器[8,89] 已将解码算法简化为父节点（一个词修饰的词）的分类，即依靠多层长短期记忆网络（LSTM）来捕捉一个词在上下文中的句法特征，就可以在宾州树库上达到 95 以上的准确率。

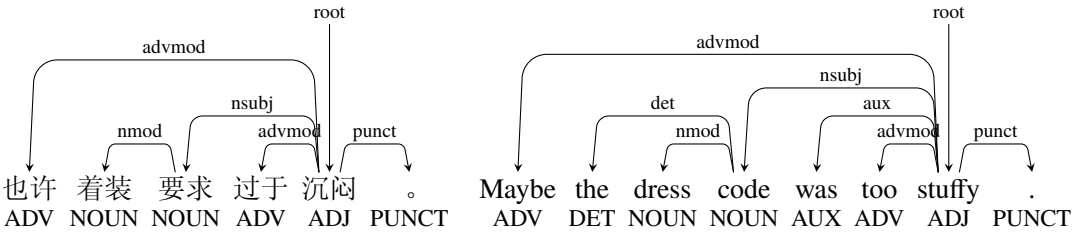

图 4-1 通用依存句法的例子。图中左右两句互为翻译，并且共现相似的句法结构。

Fig.4-1 Examples of universal dependency trees. The sentence in left and right figures are translation to each other and they share similar syntactic structure.

半监督机器学习 有效地使用未标记数据一直是半监督依存句法分析研究的热点。在深度学习广泛得到应用之前，Chen 等人在 2009 年的文献 [124] 中提出使用从自动分析的大规模未标注数据中抽取特征来提高基于图的依存句法分析的性能。这一工作在 Chen 等人 2015 年的文献 [148] 中得到进一步的发展。Liang 等人在 2008 年的文献 [92] 中也研究了如何以未标注数据为桥梁，用简单的模型近似复杂的模型。由于词向量是从大规模未标注数据中学习的，在神经网络中使用词向量的方法也可以视作是半监督学习的成功应用。上下文相关词向量[13,16] 沿着这一科研思路通过从未标注数据中学习上下文相关的词向量改进多个 NLP 任务。一系列后续工作[14,77,96] 试图揭示其成功的原因。

本章遵照使用未标注数据提高模型性能的思路开展工作。本章首次在多国语上验证 ELMo 的有效性。

上下文相关词向量的分析工作 上下文相关词向量在多种任务上的成功促使学术界对分析性能提升的原因越来越感兴趣。Peters 等人在 2018 年的文献 [77] 中研究使用其他网络代替 LSTM，他们发现诸如 Gated CNN[78] 和 Transformer[18] 这样的架构也可以有效地学习高质量的上下文相关词向量。Bowman 等人在 2018 年的文献 [149] 中从多任务学习问题的角度研究了上下文相关词向量的效果，并研究了语言建模之外的潜在获得上下文相关词向量的任务。他们发现，在研究过的任务中，语言建模仍然是最优的选择。Zhang 等人在 2018 年的文献 [150] 中进一步在句法分析上展示了语言模型相对翻译的优越性来证实这一点。Tenney 等人在 2018 年的文献 [151] 也得到了类似的结论。

4.3 深度双仿射句法分析器

本章的模型基于 Dozat 和 Manning 在 2016 年的文献 [8] 中提出的**深度双仿射**句法分析算法（deep biaffine parser）。这一算法的核心思想是使用 BiLSTM 为每个单词生成向量表示，然后使用该表示来预测词性和依存句法关系。

词性标注模型 对于词性标注这类单一输入的任务，该表示可以形式化地定义为

$$\mathbf{h}_i = \text{BiLSTM}\left(\mathbf{v}_1^{(\text{word})}, \cdots, \mathbf{v}_n^{(\text{word})}\right)_i 。 \tag{4-1}$$

其中，$\mathbf{v}_i^{(\text{word})}$ 是词表示。在获得了隐层表示 $\mathbf{h}_i$ 后，其模型依靠如下公式计算第 i 个词被标记为第 j 个词性的分数：

$$\begin{aligned}
\mathbf{h}_i^{(\text{pos})} &= \text{MLP}^{(\text{pos})}(\mathbf{h}_i) \\
\mathbf{s}_i^{(\text{pos})} &= \text{MLP}^{(\text{pos'})}(\mathbf{h}_i^{(\text{pos})}) \\
y_i^{(\text{pos})} &= \underset{j}{\text{argmax}}\, s_{i,j}^{(\text{pos})}
\end{aligned}$$

整个模型的框架如图4-2所示。

依存句法分析模型 对于依存句法分析模型，其输入包括词与词性。所以为了计算 $\mathbf{h}_i$，深度双仿射句法分析算法将两种表示进行拼接，然后输入 BiLSTM 模型中。这一过程可以形式化地描述如下：

$$\mathbf{v}_i = \mathbf{v}_i^{(\text{word})} \oplus \mathbf{v}_i^{(\text{tag})} \tag{4-2}$$

$$\mathbf{h}_i = \text{BiLSTM}(\mathbf{v}_1, \cdots, \mathbf{v}_n)_i 。 \tag{4-3}$$

在获得了 $\mathbf{h}_i$ 后，深度双仿射句法分析算法将一对词的表示输入到一个深度双仿射网络中，以求得这对词构成修饰关系的可能性。形式化地，第 i 个词修饰句子中其他词的概率可以使用如下方法进行计算

$$\begin{aligned}
\mathbf{s}_i^{(\text{arc})} &= H^{(\text{arc-head})} W^{(\text{arc})} \mathbf{h}_i^{(\text{arc-dep})} + H^{(\text{arc-head})} \mathbf{b}^{(\text{arc})} \\
y^{(\text{arc})} &= \underset{j}{\text{argmax}}\, s_{i,j}^{(\text{arc})} 。
\end{aligned}$$

其中，$\mathbf{h}_i^{(\text{arc-dep})}$ 表示第 i 个词做修饰词时的向量表示。$\mathbf{h}_i^{(\text{arc-dep})}$ 可以通过将 $\mathbf{h}_i$ 输入 MLP 计算获得。与之类似，$\mathbf{h}_i^{(\text{arc-head})}$ 代表表示第 i 个词做核心词时的向量表示。$H^{(\text{arc-head})}$ 是将 $\mathbf{h}_i^{(\text{arc-head})}$ 沿词序列方向堆叠获得的矩阵。在求得了每个词的核心词 $y^{(\text{arc})}$ 后，深度双仿射句法分析算法对每条弧依据如下公式进行分类，

$$\begin{aligned}
\mathbf{s}_i^{(\text{rel})} &= \left(\mathbf{h}_{y^{(\text{arc})}}^{(\text{rel-head})}\right)^T \mathbf{U}^{(\text{rel})} \mathbf{h}_i^{(\text{rel-dep})} + W^{(\text{rel})} \left(\mathbf{h}_i^{(\text{rel-dep})} \oplus \mathbf{h}_{y^{(\text{arc})}}^{(\text{rel-head})}\right) + \mathbf{b}^{(\text{rel})}, \\
y^{(\text{rel})} &= \underset{j}{\text{argmax}}\, s_{i,j}^{(\text{rel})} 。
\end{aligned}$$

其中，$\mathbf{h}^{(\text{rel-head})}$ 与 $\mathbf{h}^{(\text{rel-dep})}$ 的计算过程与 $\mathbf{h}_i^{(\text{arc-dep})}$ 和 $\mathbf{h}_i^{(\text{arc-head})}$ 相仿。整体的框架如图4-3所示。

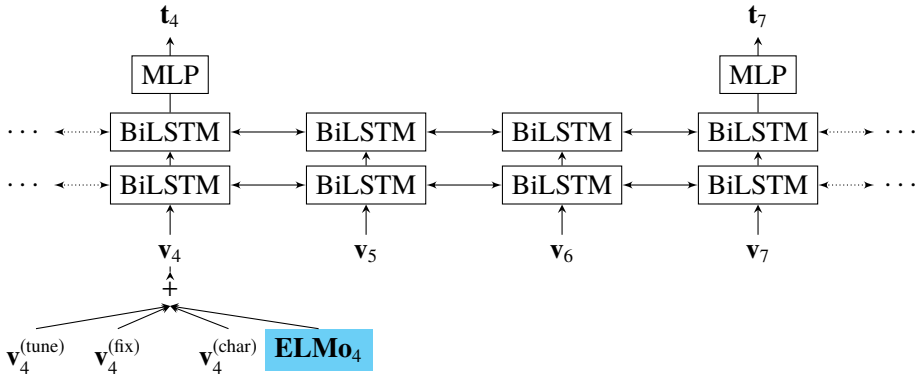

图 4-2 深度双仿射词性标注器的框架。青色的框架表示使用 ELMo 作为输入。

Fig. 4-2 The framework for deep biaffine tagger. In this paper, we use ELMo as an additional word input, as highlighted in the cyan block.

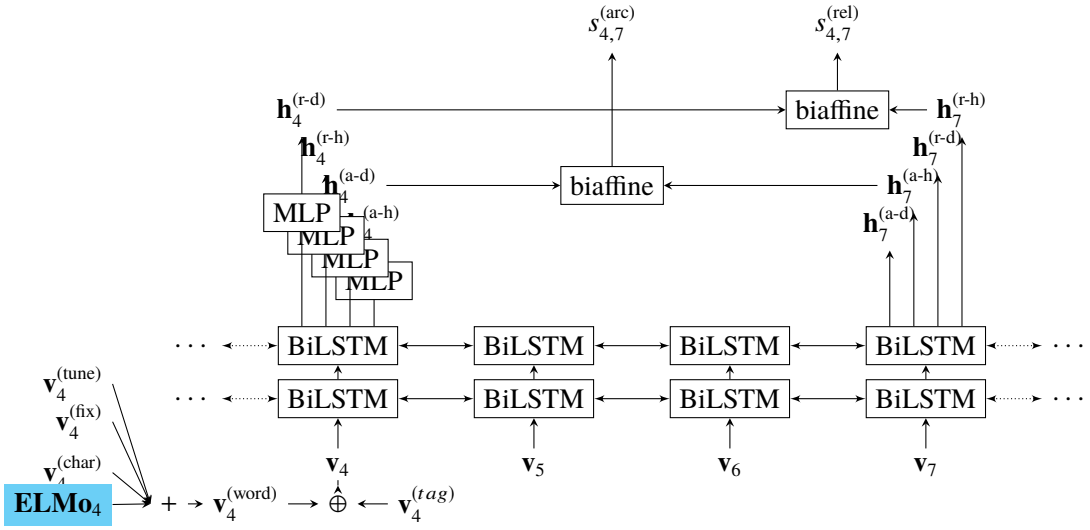

图 4-3 深度双仿射依存句法分析器的框架。青色的框架表示使用 ELMo 作为输入。该图显示了如何计算第四个词修饰第七个词的分数。

Fig. 4-3 The framework for deep biaffine parser. We demonstrate how to calculate the arc score ($s_{4,7}^{(\text{arc})}$) and relation score ($s_{4,7}^{(\text{rel})}$) between the fourth and seventh words. Like in Figure 4-2, we highlight the additional ELMo embeddings in the cyan block.

在深度双仿射句法分析算法中，由于每个词对应的核心词是独立计算的，所以求得 y^{arc} 后可能会产生环。文献 [8] 采用了一种启发式算法迭代地修改每个词的核心节点。关于算法的细节，读者可以参考原文。

输入表示 在深度双仿射句法分析算法中，最终的词向量 $\mathbf{v}_i^{(\text{word})}$ 由三部分组成：1）一个可更新的词向量 $\mathbf{v}_i^{(\text{tune})}$；2）一个固定的来自预训练的词向量 $\mathbf{v}_i^{(\text{fix})}$，以及 3）一个使用字级别 LSTM 编码的向量 $\mathbf{v}_i^{(\text{char})}$。最终的词向量由三种表示如下式加和求得

$$\mathbf{v}_i^{(\text{word})} = \mathbf{v}_i^{(\text{tune})} + \mathbf{v}_i^{(\text{fix})} + \mathbf{v}_i^{(\text{char})}。 \tag{4-4}$$

根据 Dozat 等人 2017 年的文献 [89]，可调的词向量 $\mathbf{v}_i^{(\text{tune})}$ 可以学习常见词的表示；固定的词向量 $\mathbf{v}_i^{(\text{fix})}$ 可以学习从未标注数据中无监督地学习；基于字的 LSTM 的词向量 $\mathbf{v}_i^{(\text{char})}$ 编码了词的形态学特征。而使用加和的方式表示特征借鉴了 Residual Network[74] 的思想。

4.4 基于上下文相关词向量的深度双仿射句法分析器

上下文相关词向量[12–16] 在多项句法语义任务中表现出很好的性能。Peters 等人在 2018 年的文献[13] 中提出基于语言模型的上下文相关词向量— ELMo。本文已经在第1.2.2节中对于 ELMo 的上下文表示以及模型学习进行了详细的介绍。本章在这里按照公式1-6获得上下文相关词向量 $\text{ELMo}_i = \gamma \sum_{k=0}^{L} s_k \cdot \left(\overrightarrow{\mathbf{h}}_i^{(k)} \oplus \overleftarrow{\mathbf{h}}_i^{(k)}\right)$。

在深度双仿射句法分析器中使用 ELMo　本章关注的问题是 ELMo 在通用依存句法分析中的效果。本章将 ELMo 用作一种额外的词向量输入模型中。

文献 [152] 在多个任务上研究了在使用上下文相关词向量时是否需要调整词向量参数的问题。根据其结论，在将上下文相关词向量应用于有丰富的任务相关参数的模型时固定参数可以获得更好的效果。由于本章使用的深度双仿射句法分析器包含丰富的参数，本章参考文献 [152] 并在训练模型中固定参数。

在获得 $\mathbf{ELMo}_i$ 之后，本章将其用作一个额外的词向量。正如在第4.3节中所示，文献 [89] 通过相加将不同的词特征进行组合并获得词表示（如公式4-4所示）。本章参考其做法并将 $\mathbf{ELMo}_i$ 映射到与 $\mathbf{v}^{(\text{word})}$ 相同的维度并通过相加与其他特征进行组合。因此，对于词性标注与句法分析器（参考图4-2与图4-3中青色的部分），$\mathbf{v}_i^{(\text{word})}$ 等于

$$\mathbf{v}_i^{(\text{word})} = \mathbf{v}_i^{(\text{tune})} + \mathbf{v}_i^{(\text{fix})} + \mathbf{v}_i^{(\text{char})} + W^{(\text{ELMo})} \cdot \mathbf{ELMo}_i \tag{4-5}$$

文献 [13] 探索了两类将上下文相关词向量应用于下游任务的方法。一种是将其与传统词向量进行组合，另一种是与隐层进行组合。根据文献 [13]，复杂的组合方法带来了少量性能的提升。然而，由于本章的目标是研究上下文相关词向量与通用依存分析的关系，本章只使用将上下文相关词向量与传统词向量进行组合的方式在深度双仿射句法分析器中使用 ELMo。

4.5 实验

4.5.1 设置

本章在 CoNLL 2018 多国语句法分析评测[146] 中的 57 个数据集上进行实验。这些实验包括通用词性标注和通用依存句法分析。这 57 个数据集涵盖 44 种不同

语系的语言。本章选择这些数据集的原因是每个数据集对应的语言都有与之配套的生语料。本章采用标准方法将数据集划分为训练集、开发集和测试集。为了训练 ELMo，本章从评测公开的生语料中为每种语言随机抽取了两千万词的生语料。

对于通用词性标注实验，本章使用正确的分词结果作为输入。评测任务为词性标注提供了通用词性标签（UPOS）与每个树库特有的词性标签（XPOS）。本章参考文献 [89] 并使用 UPOS 与 XPOS 联合学习的方法训练本章的词性标注器。具体来讲，联合学习的目标是在使用共享的的句子表示的同时正确预测 UPOS 和 XPOS。本章词级别词性标注准确率作为本章的评价指标。由于某些数据集的 XPOS 无意义[①]，文本只报告 UPOS 结果。初步实验结果显示了 UPOS 和 XPOS 的类似趋势，故报告 UPOS 结果具有足够的代表性。

对于通用依存句法分析实验，与词性标注类似，本章也使用正确的分词结果作为输入。为了使训练过程与测试过程相近，本章使用自动词性标注作为输入。为了获得训练数据上的自动词性，本章在词性标注实验中的词性标注器的基础上，利用 5 折交叉方法给整个训练集标注了自动词性。本章使用带标签依存关系准确率（LAS）评估通用依赖性分析性能。

训练 ELMo 的一个困难之处在于：当词表较大，softmax 输出层的维度过高。为了使训练可行，本章使用 sampled softmax 技术[80] 训练模型。本章的训练方法和标准 sampled softmax 之间的主要差异在于本章使用目标词的一定窗口内的词作为负样本。[②] 初步实验表明，这种方法有更好的性能。在 NVIDIA P100 GPU 上，训练一种语言的 ELMo 大约需要 3 天。

4.5.2 结果

表 4-1: 在通用依存句法分析中使用 ELMo 的比较。
Table 4-1: Comparison of model with and without ELMo.

	未使用 ELMo		使用 ELMo	
树库	UPOS	LAS	UPOS	LAS
Arabic-PADT	96.73	82.88	**96.93**	**83.64**
Bulgarian-BTB	98.40	90.80	**99.13**	**91.78**
Catalan-AnCora	97.37	91.05	**98.89**	**91.40**
Czech-CAC	99.01	91.22	**99.25**	**91.40**
Czech-FicTree	98.55	91.08	**98.57**	**91.57**
Czech-PDT	98.99	92.36	**99.13**	**92.39**
Old_Church_Slavonic-PROIEL	**96.61**	81.01	96.47	**82.26**
Danish-DDT	96.97	85.19	**97.95**	**86.34**

① 例如 Japanese-GSD 只提供了 UPOS 标注，XPOS 全部为占位符。
② 本章使用的窗口大小为 8,172。

表 4-1（续表）

树库	未使用 ELMo		使用 ELMo	
	UPOS	LAS	UPOS	LAS
German-GSD	94.40	81.26	**94.89**	**81.79**
Greek-GDT	96.57	88.18	**98.02**	**89.31**
English-EWT	95.96	87.62	**96.74**	**88.27**
English-GUM	93.57	84.08	**96.42**	**86.00**
English-LinES	96.52	79.68	**97.14**	**80.46**
Spanish-AnCora	97.78	90.44	**98.84**	**90.86**
Estonian-EDT	96.26	85.10	**97.49**	**85.48**
Basque-BDT	95.22	81.85	**96.48**	**83.56**
Persian-Seraji	97.22	87.20	**97.93**	**88.15**
Finnish-FTB	95.89	89.11	**96.87**	**89.94**
Finnish-TDT	96.24	87.99	**97.48**	**89.65**
French-GSD	97.13	87.93	**97.59**	**88.62**
French-Sequoia	98.27	89.12	**98.91**	**91.18**
French-Spoken	94.24	74.95	**96.31**	**77.79**
Galician-CTG	**97.71**	83.31	**97.71**	**83.45**
Ancient_Greek-Perseus	92.05	75.12	**94.39**	**78.73**
Ancient_Greek-PROIEL	96.86	81.76	**97.85**	**83.75**
Hebrew-HTB	96.43	85.21	**97.34**	**87.21**
Hindi-HDTB	97.24	92.00	**97.60**	**92.19**
Croatian-SET	97.35	86.51	**98.14**	**87.24**
Hungarian-Szeged	92.96	76.69	**96.59**	**81.35**
Indonesian-GSD	91.82	79.21	**93.76**	**79.30**
Italian-ISDT	97.89	91.52	**98.41**	**92.48**
Italian-PoSTWITA	95.93	80.54	**96.82**	**81.77**
Japanese-GSD	97.76	92.51	**97.96**	**92.70**
Korean-GSD	96.25	83.71	**96.60**	**84.94**
Korean-Kaist	95.32	**86.62**	**95.68**	86.40
Latin-ITTB	**98.55**	**89.04**	98.53	88.90
Latin-PROIEL	96.23	79.06	**97.03**	**80.49**
Latvian-LVTB	93.96	82.43	**96.07**	**83.87**
Dutch-Alpino	96.09	87.43	**96.35**	**90.21**
Dutch-LassySmall	95.78	85.96	**96.61**	**87.51**
Norwegian-Bokmaal	97.46	90.78	**98.36**	**91.67**
Norwegian-Nynorsk	97.19	90.42	**98.20**	**91.33**
Polish-LFG	97.90	**95.03**	**98.83**	94.73
Polish-SZ	97.24	91.74	**98.32**	**92.18**
Portuguese-Bosque	96.20	89.12	**97.21**	**89.85**
Romanian-RRT	97.53	86.90	**97.97**	**86.95**
Russian-SynTagRus	97.81	92.96	**99.02**	**93.09**
Slovak-SNK	93.96	88.11	**97.08**	**90.13**
Slovenian-SSJ	97.74	92.01	**98.61**	**92.04**
Swedish-LinES	96.39	81.72	**97.38**	**83.21**
Swedish-Talbanken	97.60	86.61	**98.09**	**88.26**
Turkish-IMST	95.66	64.97	**96.65**	**68.03**

表 4-1（续表）

树库	未使用 ELMo UPOS	未使用 ELMo LAS	使用 ELMo UPOS	使用 ELMo LAS
Uyghur-UDT	**89.84**	65.32	89.68	**68.13**
Ukrainian-IU	96.20	85.50	**97.74**	**86.40**
Urdu-UDTB	**94.39**	81.66	94.21	**82.26**
Vietnamese-VTB	88.99	60.27	**90.54**	**62.12**
Chinese-GSD	**94.60**	79.70	94.30	**79.85**

表4-1通用词性标注和通用依存句法分析的性能。表4-1显示 ELMo 给 57 个树库带来稳定的的性能提升。具体地说，ELMo 给 57 个树库中的 51 个的树库提高了 UPOS 的准确率。平均准确率提升为 0.91，词性标注标记的平均误差（error reduction）降低了 24‰。对于通用依存句法分析，ELMo 给 57 个树库中的 54 个树库带来 LAS 的准确率的提升。平均 LAS 提升为 1.11，平均误差降低了 7‰。上面的结果表明 ELMo 可以有效提高通用词性标注和通用依存句法分析的性能。

本章也在表4-2中将本章的使用 ELMo 的模型与当前最优的通用句法分析系统进行了对比。对比的系统包括；1）词性句法联合模型 UDpipe 2.0[153]；2）多语言多任务联合模型 UDify[154]。这一比较显示 ELMo 能够帮助通用词性标注/依存句法分析取得当前最优的性能。

本章还研究了 UPOS 提升与 LAS 提升之间的相关性。其 Pearson 相关系数为 0.5196（$p < 0.001$）。这表明使用 ELMo 所带来的改进在两个语法任务中是一致的。

4.6 分析

4.6.1 树库属性与性能提升的关系

第4.5.2节显示 ELMo 可以给通用词性标注与通用句法分析带来性能的提升。然而，对于不同的树库，提升的幅度是不同的。ELMo 给 Hungarian-Szeged 带来的提升幅度最大，但给 Galician-CTG 带来了负面效果。① 这一观察表明 ELMo 的作用可能与树库的某些属性相关。为了深入了解这些属性，本章对不同语言的树库进行分析实验，检查他们的属性是如何与性能提升相关的。本章关注一个树库五方面的属性，这些属性包括：

- 训练数据大小：本章使用训练数据中的词数来表示训练大小。②
- 形态学丰富程度：本章按照文献 [155] 建议，使用语料库中的词熵作为该语料库的形态学丰富程度的评估。根据文献 [155]，单词熵与语言学家定义的形态复

① ELMo 给 *Galician-CTG* 带来了 0.30 的 UPOS 下降，给 *Hungarian-Szeged* 带来了 4.56 的 UPOS 提升。
② 以 10 为底的对数。

表 4-2 本章使用 ELMo 模型与当前最优通用句法分析系统的对比。
Table 4-2 The comparison with the state-of-the-art UD parsing systems.

模型	宏平均-UPOS	宏平均-LAS
UDpipe 2.0[153]	96.94	85.05
UDify[154]	95.88	85.50
深度双仿射（未使用 ELMo）	96.15	84.94
深度双仿射（使用 ELMo）	**97.07**	**86.04**

杂性①有很好的相关性。

• 非投射程度：本章使用训练数据中非投影弧的百分比作为非投射程度的评价。

• 多义词丰富程度：本章将拥有多个词性的词近似定义为多义词。本章使用树库中多义词的比例作为树库的多义词丰富程度。由于上下文相关词向量的设计初衷是解决一词多义问题，本章希望通过这一指标衡量上下文相关词向量能否给多义词丰富的树库带来更大的提升。

• 未登录词比例：本章将在测试数据出现但不在训练数据出现的词定义为未登录词（OOV）。OOV 的计算与生语料无关。本章将测试数据中此类单词的百分比定义为未登录词比例。

其中训练数据大小、形态学丰富程度以及非投射程度在文献 [89] 也得到了讨论。

本章首先在图4-4中显示了本章分析的树库属性与词性/句法性能的绝对提升/错误率降低之间的关系。除了非投射性，图4-4显示这些属性与性能提升基本单调（monotonic）地相关。基于这一图示，本章使用两种统计相关性评价方法---评价两个变量线性相关性的 Pearson 相关系数和评价两个变量排序相关性的 Spearman 相关系数。表4-3显示了相关性的评价结果由于相关系数的绝对值不具备可解释性，本章只使用这个值衡量哪个属性与性能提升更相关。表4-3显示在所有这些因素中，未登录词比例与通用词性标注性能提升的相关性最大。训练数据大小与通用句法分析性能提升的相关性最大。而未登录词比例与通用句法分析性能提升的相关性接近。需要强调的是，训练数据较少的树库，通常会有更多的未登录词出现在测试数据中。② 因此，本章认为未登录词比例是一个更相关并且信息丰富的属性。

除了这些树库相关属性，ELMo 的带来的性能提升与非树库属性存在潜在的相关性。为了探索这种相关性，本章参考 Ma 等人在 2018 年的文献 [157] 以及 McDonald 与 Nivre 在 2007 年的文献 [158] 并研究了三个结构相关的属性。这些属

① 参考：世界语言形态学项目 [156]

② 在本章实验的数据集中，训练数据大小和未登录词比例之间的皮尔逊相关性为-0.5477。

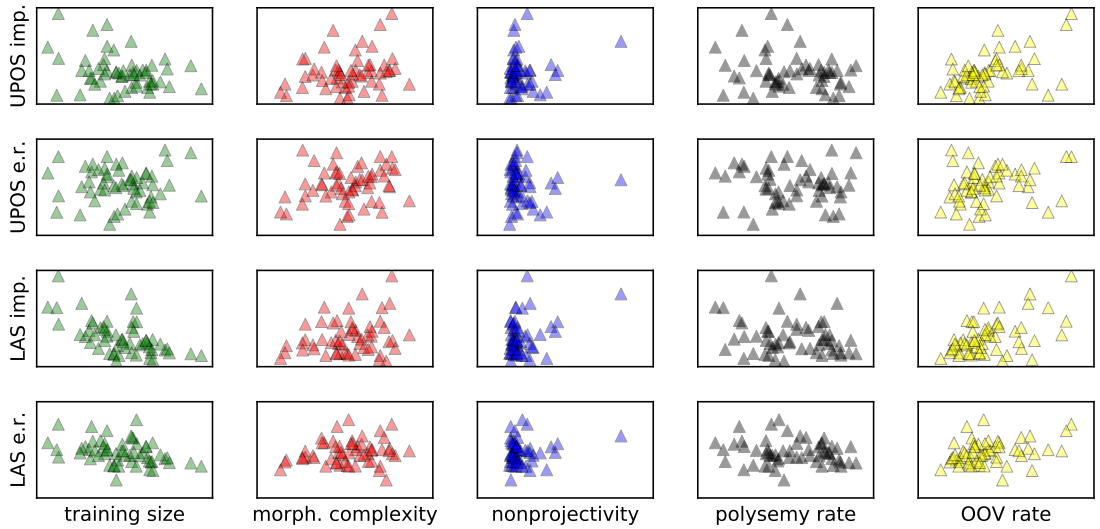

图 4-4 五种树库属性与词性/句法准确率提升绝对值 (imp.) 与错误率降低 *(e.r.)* 的关系，图中的每个点代表一个树库。

Fig. 4-4 The relation between five attributes and the ELMo's improvements (*imp.*) and error reduction (*e.r.*) for POS tagging (*UPOS*) and UD parsing (*LAS*). Each point in this figure denotes a treebank.

表 4-3 树库性质与性能提升的 Pearson/Spearman 相关性。粗体表示相关性最高的性质。

Table 4-3 The Pearson/Spearman correlation between five attributes and the improvements and error reduction for each treebank. Bold number show the highest (absolute) correlation value.

	UPOS		LAS	
树库属性	提升	误差降低	提升	误差降低
训练数据大小	-0.30 / -0.23	0.03 / -0.05	**-0.58 / -0.56**	**-0.36 / -0.39**
形态学丰富程度	0.26 / 0.25	0.20 / 0.20	0.24 / 0.19	0.17 / 0.17
非投射程度	0.14 / 0.05	-0.07 / -0.09	0.33 / 0.11	0.14 / 0.04
多义词丰富程度	-0.20 / -0.15	-0.10 / -0.15	-0.17 / -0.14	-0.13 / -0.12
未登录词比例	**0.51 / 0.44**	**0.22 / 0.24**	0.52 / 0.43	0.33 / 0.33

性包括弧长度（dependency length）、深度（distance to root）和修饰词个数（number of modifier siblings)。本章研究这些属性的一个出发点是使用 ELMo 的模型可能为某种类型的弧带来更大的提升。比如使用 ELMo 的模型利用 ELMo 更好地编码上下文，从而取得更好的长距离弧的性能。本章在图4-5中显示了这一分析结果。图4-5中也包含使用 ELMo 给每种结构属性带来的错误率降低。可见，不同指标下的使用 ELMo 带来的错误率降低几乎一致。这一观察显示了结构属性与性能提升没有显著相关性。因此，本章在后文分析中着重关注 ELMo 在未登录词方面的性能。

4.6.2 未登录词的性能

为了进一步证实 ELMo 带来的改进主要发生在未登录词中，本章在合并后的

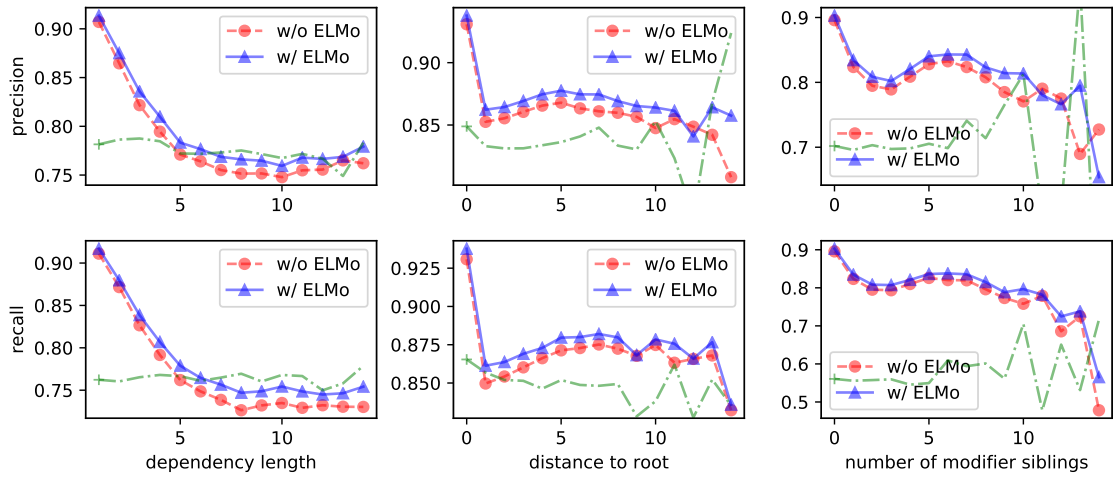

图 4-5 依存弧三种属性对应的准确率/召回率。绿色虚线代表错误率降低，错误率降低的范围在 0.0 到 0.3。

Fig. 4-5 Dependency arc precision/recall relative to three attributes. The dashed green line shows the error reduction. The limits of y-axis for error reduction are fixed to (0.0, 0.3).

表 4-4 使用与未使用 ELMo 模型在登录词与未登录词上的表现。最后一行显示了误差减少。

Fig.4-4 A comparison on the IV and OOV performance of two models. The last row shows the error reductions.

	登录词		未登录词	
模型	UPOS	LAS	UPOS	LAS
未使用 ELMo	98.00	87.97	88.50	79.52
使用 ELMo	**98.27**	**88.58**	**93.36**	**81.32**
	13.50%	5.07%	42.26%	8.79%

测试数据上比较了模型的在登录词和未登录词的性能。[①] 表4-4显示了比较的结果。根据表4-4，ELMo 在通用词性标注和通用依存句法分析上给未登录词带来了比词典内词更多的提升。对于通用词性标注，提升的绝对值为 4.86，误差减少为 42.26%。对于通用依存句法分析，提升的绝对值为 1.80，误差减少为 8.79%。

除了未登录词的详细性能，本章还在图4-6中绘制了使用与不使用 ELMo 的模型的通用词性标注误差分布。通过图4-6可以看出：两类模型在如助词（AUX），并列连词（CCONJ）和从句连词（SCONJ）一类的功能词上表现几乎一致。但是，使用 ELMo 模型明显表现出更好的专有名词（PROPN）性能。这个结果从另一种角度证明了 ELMo 对 OOV 的影响。因为作为一个新兴的实体名称，未登录词中专有名词的比例更大。

至此，本章的一系列分析都指出：**ELMo 主要通过提高未登录词的性能来提高语法 NLP 任务的性能。**

① IV 和 OOV 词的比例测试数据为 87.38%和 12.62%。

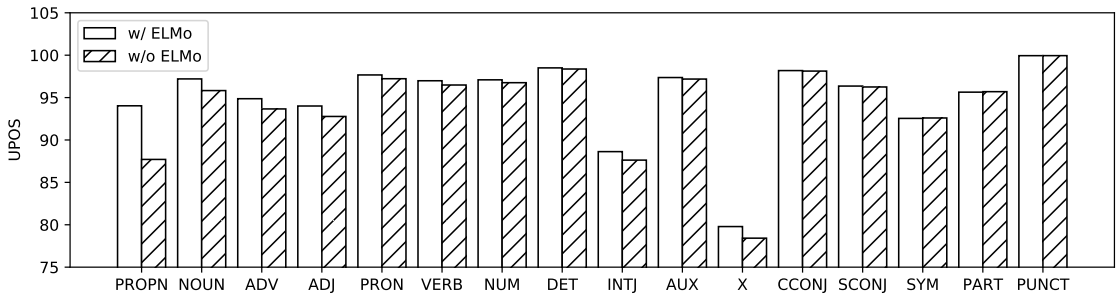

图 4-6 使用与未使用 ELMo 模型对应的词性标注的错误分布。

Fig. 4-6 The POS tag error distributions of our models with and without ELMo.

4.6.3 未登录词句法抽象能力的分析

未登录词一直是影响 NLP 系统性能的问题。未登录词问题的一般解决方案是用高层次抽象来表示单词。例如，Collins 在 2002 年的文献 [37] 中使用前缀和后缀功能来表示词性标注任务中的单词。Liang 在 2005 年文献 [159] 在解决词性标注问题时使用词聚类作为抽象的词性。Ballesteros 等人在 2015 年的文献 [160] 中使用字符级 LSTM 替换词向量，从而在基于转移的依存句法分析中取得了更好的效果。在学习更好的词的抽象方面，ELMo 是一个潜在的解决方案，原因有二：1）ELMo 的训练过程遇到很多单词，这意味着 ELMo‘‘记忆’’ 单词。2）ELMo 使用根据上下文表示（BiLSTM）以及词形表示（CNN）对单词的句法功能进行更好的 ‘‘抽象’’。

直观地讲，‘‘记忆’’ 效果可以为未登录带来性能提升。然而，由于在学习 ELMo 时穷尽所有词汇是不现实的，本章更加关注需要 ‘‘抽象’’ 的未登录词的性能。为了评价需要 ‘‘抽象’’ 的未登录词的性能，本章将未登录词分为两类，代表在学习 ELMo 时是否遇到（召回）这个词。在召回的未登录词上的表现可以被视为对 ‘‘记忆’’ 效果的一种评估，而未召回的未登录词则反映了 ELMo 的 ‘‘抽象’’ 效果。表4-5 中显示了通用词性标注和通用依存句法分析上这两类词的性能。根据表4-5，使用 ELMo 的模型为不同（召回与未召回）的未登录词都带来了性能的提升，而且两类未登录词的提升相近。这个结果表明 ELMo 既能记忆未登录词，也能抽象未登录词。未登录词的抽象能力是 ELMo 性能提升的重要来源。

4.6.4 未登录词词形抽象能力的分析

根据前文，ELMo 具备抽象一个词的句法功能的能力。本章希望进一步探究这类抽象能力的来源。如前文讨论，未登录词的句法功能可以通过 1）它的上下文以及 2）它的词形进行推断。为了判断这两种因素哪种贡献更大，本章对于模型输入的不同组合进行了研究。为了消除深度双仿射分析器中字级别表示（公式4-4中的 $\tilde{\mathbf{v}}$）的影响，本章只使用可训练的词向量 $\mathbf{w}$ 与固定词向量 $\mathbf{p}$ 的组合作为词向量。其中固定词向量包括：

表 4-5 在 ELMo 学习过程中召回（出现在 ELMo 的训练数据中）与未召回（未出现在 ELMo 的训练数据中）的两类未登录词的性能的比较。

Table 4-5 A comparison on the OOV word performance. We categorize OOV words into two types, indicating whether a word is recalled during the learning of ELMo.

	召回		未召回	
模型	UPOS	LAS	UPOS	LAS
未使用 ELMo	88.85	79.52	88.29	79.52
使用 ELMo	**93.90**	**81.42**	**92.05**	**80.99**
	45.29%	9.27%	32.11%	7.18%

表 4-6 消融实验结果。括号里的数代表未召回的未登录词的准确率。

Table 4-6 The ablation results. The number in the brackets shows the performance of unrecalled OOV.

	UPOS			LAS		
	宏平均	登录词	未登录词	宏平均	登录词	未登录词
Word2vec	93.04	96.80	72.59 (62.83)	82.73	84.94	72.15 (62.83)
FastText	94.91	96.81	83.69 (75.05)	82.83	84.73	73.34 (68.23)
基于词的 ELMo	95.64	97.42	85.52 (68.48)	84.06	85.96	74.68 (64.07)
ELMo-CNN	96.60	97.58	90.74 (85.56)	84.75	86.19	77.38 (72.79)
基于字的 ELMo	**96.82**	**97.68**	**91.50 (86.74)**	**84.94**	**86.43**	**77.82 (73.60)**

- *Word2vec*：Word2vec 是本章的未使用上下文相关词向量的基线实验系统。
- *FastText*：FastText[4] 是一种能够建模任意词的静态词向量。由于 FastText 具备建模任意词的能力，本章将 Word2vec 系统中的 Word2vec 替换为 FastText 以验证提高未登录词建模能力是否能够带来性能提升。
- *ELMo-CNN*：本章使用 ELMo 中上下文无关的词向量（CNN 输出）替代 ELMo。这一模型移除了 ELMo 的上下文建模能力，从而直接讨论词形抽象对于性能的影响。
- 基于词向量的 *ELMo*：本章将 ELMo 中的 CNN 替换为普通词向量并按照第4.5.1节的方法重新训练了 ELMo。这一模型移除了 ELMo 的词形抽象能力，从而直接讨论建模上下文对于性能的影响。
- 基于字的 *ELMo*：本章的使用上下文相关词向量的模型。

本章在六种语言①上进行这一实验。实验结果如表4-6所示。根据表4-6，使用基于字级别词表示的模型（*ELMo-CNN* 和基于字的 *ELMo*）的性能显著好于使用传统词向量的模型（*Word2vec* 和基于词的 *ELMo*）。而使用基于字级别词表示的模型带来了显著的未登录词的错误率降低。这一实验结果表明基于字级别词表示在 ELMo 的词抽象能力中发挥了很大的作用。

① 包括荷兰语、英语、法语、匈牙利语、意大利语和斯洛伐克语。

表 4-7 使用 ELMo 的模型对于多义词/单义词的句法分析性能。本章将包含多个词性的词近似定义为多义词。最后一行显示错误率降低。

Table 4-7 A comparison of the polysemous/monosemous word performance. We determine a word as polysemy when it has multiple POS in the training data. The last row shows the error reductions.

模型	多义词	单义词
未使用 ELMo	88.90	87.84
使用 ELMo	**89.47**	**88.12**
	5.13%	5.13%

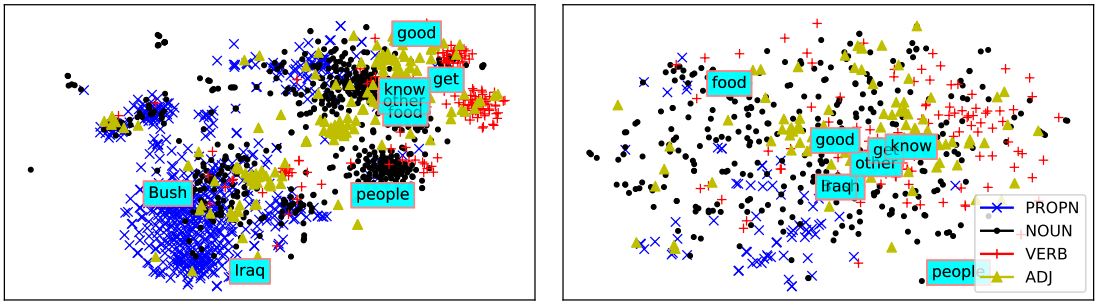

图 4-7 未登录词的可视化的结果。左图显示的是 ELMo 上下文无关层的结果，右图显示的是 Word2vec 的结果。图中也包含四种词性的典型词。由于 Word2vec 采用固定词表的原因，无法给每个未登录词以适当的表示。所以左侧比右侧有更多的点。

Fig. 4-7 Visualization of the OOV word embeddings from ELMo (left) and Word2vec (right) along with the prototype word for 4 POS tags for content words. The left figure presents more words than the right one because ELMo can produce embeddings for any word, while Word2vec only yields embeddings for encountered words, and some OOV words are not recalled.

通过比较表4-6中 *Word2vec* 与 *FastText* 的结果，使用 *FastText* 作为输入的模型的性能好于 *Word2vec*。这一结果表明建模词形信息能够提升模型性能。然而，使用 *FastText* 的模型相较使用字级别词表示的模型仍有较大差距，可见 ELMo 的字级别词表示具有更强的表示能力。本章猜测这种表示能力来自 CNN 的建模字 n-gram 的网络结构以及语言模型学习目标。

另一个值得注意的结果是字级别词表示（*ELMo-CNN*）取得了与完整 ELMo 相近的性能。同时，这一结果显著好于基于词的 ELMo。由于基于词的 ELMo 只建模了上下文，为了进一步观察建模上下文对于模型性能的影响，本章将词依据是否具有多个词性近似分类为多义词与单义词。单义词与多义词在句法分析上的性能如表4-7所示。根据表4-7的结果，单义词与多义词的错误率降低几乎相同。本章猜测，尽管 ELMo 的设计初衷是建模一词多义，在将其应用于句法分析时，其作为副产物产生的上下文无关的表示反而发挥了更大的作用。

前文将模型性能的提升归因于 ELMo 在可以学习更好的未登录词的的句法抽

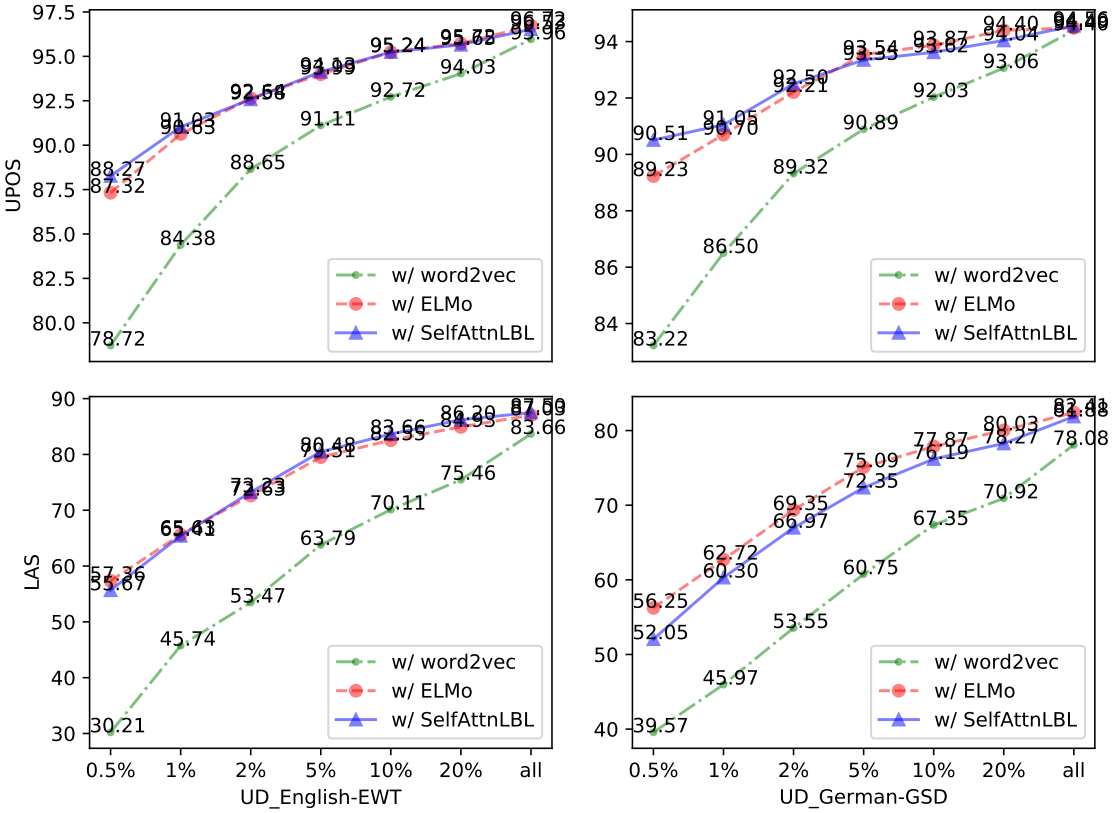

图 4-8 数据稀缺句法分析的实验结果

Fig. 4-8 The simulation results on low-resource settings

象。为了通过实证测试，本章对 ELMo 中上下文无关的词向量（CNN 的输出）进行了可视化。本章使用 T-SNE[130] 对未登录词以及特定词性的典型词进行从高维空间到 2 维平面的可视化。① 可视化的结果显示在图4-7中。点的颜色代表相应单词的词性标签。② 这个图显示不同的未登录词的表示的聚类与他们的词性有很好的相关性，即 ELMo 产生的词表示具有很好的句法属性的抽象能力。相比之下，Word2vec 的产生的词向量分散在嵌入空间中的不同位置，不能充分捕捉单词之间的句法相似性。

4.6.5 数据稀缺句法分析模拟实验

前文展示了 ELMo 通过学习更好的未登录词抽象来提高语法分析性能。对于训练数据不足的领域，未登录词的问题往往非常严重。ELMo 乃至上下文相关词向量可以为解决这类问题的提供有效的数据支撑。为了验证这一假设，本章进行了数据稀缺句法分析模拟实验。本章随机采样了 UD_English-EWT 以及 UD_German-GSD 原训练数据中的 0.5%，1%，2%，5%，10% 以及 20% 作为训练集，并验证

① 本章关注的未登录词是从 UD_English-EWT 中提取的。本章使用 Bush 和 Iraq 作为专有名词的典型词，people 和 food 作为名词的典型词，good 和 other 作为形容词的典型词，know 和 get 作为动词的典型词。

② 约 97%的未登录词只有一个关联的 POS 标签。

了 ELMo 以及本章第2章提出的 SelfAttnLBL。实验结果如图4-8所示。根据图4-8的结果，使用上下文相关词向量可以显著提高模型在少量数据时的性能（0.5% 以及 1% 的训练数据）。ELMo 与本章提出的 SelfAttnLBL 性能相当。这一观察验证了前文的假设。根据 Howard 与 Ruder 在 2018 年的文献 [14]，预训练通过将表示能力转移到特定任务上帮助数据稀缺任务。本章进一步将这种能力细化到预训练模型对于词的抽象上。

4.7 本章小结

本章使用 ELMo 作为额外的词向量提高了通用依存句法分析的性能。本章进一步分析了性能提升的原因并将其归纳为 ELMo 为未登录词提供了更好的抽象。其中 ELMo 建模词形的能力为未登录词的良好抽象作出了重要贡献。这些结果表明上下文相关词向量是一种解决数据不足自然语言处理任务的很有前途的手段。本章也通过模拟实验验证了这一假设。

第 5 章 基于知识蒸馏的依存句法分析加速方法

5.1 引言

实现又快又好的语言分析是相关研究的重要目标，但‘‘快’’与‘‘好’’往往难以调和。准确率高的模型（例如使用上下文相关词向量的模型）往往较复杂，参数多，运行速度较慢。速度快的模型往往较简单，参数少，准确率低。知识蒸馏[161]是一类使用简单模型近似复杂模型的机器学习思想，其目标是用简单模型拟合复杂模型的输出，从而使速度快的模型具备准确率高的模型的性能。近年来，图像领域的知识蒸馏有较多研究，但语言分析，特别是语言分析中**结构预测**的知识蒸馏的相关研究相对较少。

基于搜索的结构预测将自然语言结构（如词性序列，句法树，翻译，语义图等）的生成过程建模为一个搜索问题[67,162–166]。由于较高的准确率与较快的运行速度，基于搜索的结构预测近年来获得了很多关注。基于搜索的结构预测依靠一个概率化的策略函数来指导搜索过程。这个策略函数通常通过模仿参考策略进行学习。这种模仿的过程包括训练一个分类器，使其能够在参考策略遇到的状态上预测出参考策略的搜索动作。这种模仿方式有时候是有问题的。其中一个问题是参考策略是有歧义的，即不同的搜索动作都可以达到正确的结构。但在模仿过程中，只有一种动作被用作正确实例来训练分类器[167]。另一个问题是训练测试不一致的问题，即训练过程是在正确的搜索状态中完成的，但在测试过程中，学到的策略可能会进入不正确的搜索状态[168,169]。这些问题都影响了基于搜索的结构预测的泛化性以及性能。

前人工作从两个方面解决这一问题。为了解决训练数据的歧义问题，文献 [170] 使用集成学习。为了解决训练测试不一致的问题，前人工作[166–169,171] 在训练过程中鼓励模型探索错误的状态。本章提出使用知识蒸馏[161] 的手段统一地解决这两类问题。本章提出从一个由不同初始化的模型进行集成获得的复杂模型中蒸馏出一个简单模型。蒸馏的过程可以描述为用简单模型在参考策略的状态上匹配复杂模型的概率输出。除此之外，本章也在学习过程中允许集成模型探索搜索空间，并在探索到的状态中学习模型。将这两种蒸馏过程进行结合可以进一步提高模型的性能。本章提出的模型的流程图如图5-1所示。

本章在基于搜索的结构预测问题 — 基于转移的依存句法分析上进行了实验。实验的结果均超过了强基线系统的结果，这显示了本章提出方法的有效性。在依

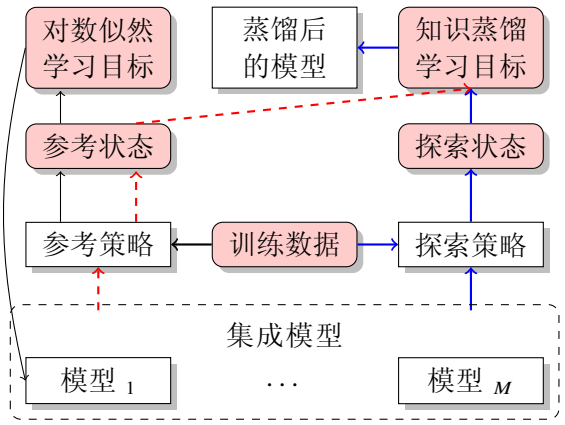

图 5-1 本章提出的基于转移的结构预测的框架图。虚线方框内的部分代表本章使用不同初始化训练的集成模型。红色虚线表示从参考状态蒸馏模型（第5.3.2节）。蓝色实线部分表示从探索状态中知识蒸馏（第5.3.3节）。

Fig. 5-1 Workflow of our knowledge distillation for search-based structured prediction. The yellow bracket represents the ensemble of multiple models trained with different initialization. The dashed red line shows our *distillation from reference* (§5.3.2). The solid blue line shows our *distillation from exploration* (§5.3.3).

存句法分析的实验中，相对基线系统的提升是 1.32。

除了从同构的集成模型中进行知识蒸馏，本章也将本章的技术应用于异构模型的知识蒸馏中。本章将前文的技术应用于对使用上下文相关词向量的模型的蒸馏。实验结果表明本章提出的方法可以有效从异构模型中进行知识蒸馏。这一结果使得不使用上下文相关词向量的模型逼近使用的模型的性能成为可能。

本章的主要贡献如下：

• 本章研究基于搜索的结构化预测中的知识蒸馏问题，并提出通过拟合参考状态（§5.3.2）和探索状态的分布来将集成模型的知识蒸馏到单模型上。从组合两种状态中学习可以进一步提升性能（§5.3.4）。

• 本章在两个基于搜索的结构预测问题 — 基于转移的句法分析以及神经机器翻译上验证了模型的性能。根据本章实验，使用知识蒸馏能够有效提高基于搜索的结构预测的性能，本章模型相较基线系统获得了显著的性能提升，同时性能优于其他基于贪心解码的结构预测模型（§5.5.2）。同时，通过全面的分析，本章经验性证明本章方法的合理性（§5.5.3）。

• 基于上述结果，本章将本章提出的知识蒸馏的技术应用于异构模型的知识蒸馏上。本章将使用上下文相关词向量的模型蒸馏到不使用的模型上（§5.5.2），并获得了相近的性能。这一结果为加速使用上下文相关词向量的模型提供了思路。

5.2 背景知识

5.2.1 基于转移的依存句法分析

结构预测模型将输入 $\mathbf{x} = (x_1, x_2, ..., x_n)$ 映射到对应的结构输出 $\mathbf{y} = (y_1, y_2, ..., y_m)$。其中，$\mathbf{y}$ 的变量之间相互依赖。基于搜索的结构预测[38,162,168,169,172–175]将结构的生成过程建模为一个搜索过程。这个搜索过程可以形式化定义为一个 5 元组 $(\mathcal{S}, \mathcal{A}, \mathcal{T}(s,a), \mathcal{S}_0, \mathcal{S}_T)$。其中 $\mathcal{S}$ 是搜索状态的集合；$\mathcal{A}$ 是搜索动作的集合；$\mathcal{T}$ 是转移函数，这个转移函数根据输入状态和动作产生新的搜索状态，即 $\mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$；$\mathcal{S}_0$ 是初始状态的集合；$\mathcal{S}_T$ 是终结状态的集合。基于搜索的结构预测算法从某个初始状态开始 $s_0 \in \mathcal{S}_0$，不断地根据一个策略 $\pi(s)$，选择一个动作 $a_t \in \mathcal{A}$ 并将其应用在当前状态 s_t 上，从而根据 $s_{t+1} \leftarrow \mathcal{T}(s_t, a_t)$ 进入新的状态 s_{t+1}，直到达到一个终结状态 $s_T \in \mathcal{S}_T$。很多自然语言处理问题可以建模为基于搜索的结构预测问题。这些问题包括依存句法分析[26] 和神经机器翻译[67,163]。表5-1展示了使用基于搜索的结构预测建模基于转移的句法分析的方法。

在数据驱动的情景下，$\pi(s)$ 控制整个的搜索过程并且通常被建模成一个分类器 $p(a \mid s)$。这个分类器输出状态 s 下采用一个动作 a 的概率。在这种定义下，贪心搜索可以形式化地定义为根据 $p(a \mid s)$ 选择概率最高的动作，即 $\pi(s) = \text{argmax}_a\, p(a \mid s)$。为了学习最优分类器，基于搜索的结构预测需要定义一个参考策略 $\pi_{\mathcal{R}}(s, \mathbf{y})$。这个参考策略根据正确结构 $\mathbf{y}$ 给输入状态 s 一个参考动作 a。学习 $p(a \mid s)$ 的过程进而被建模为以 s 为输入，a 为标准输出的分类器学习问题。算法5-1显示了学习 $p(a \mid s)$ 的一般算法。其中包括：首先，根据 $\pi_{\mathcal{R}}(s, \mathbf{y})$ 在训练数据上产生参考状态与参考动作（算法5-1的第1到第11行）；然后，使用参考状态与参考动作以对数似然为学习目标训练 $p(a \mid s)$（算法5-1的第12行），即

$$\mathcal{L}_{NLL} = \sum_{s \in D} \sum_{a} -\mathbb{1}\{a = \pi_{\mathcal{R}}\} \cdot \log p(a \mid s)。 \tag{5-1}$$

参考策略有时是次优或者有歧义的。即在某些状态下，可能有多个动作是正确的或者说可以产生正确的结构。在基于转移的结构预测中，Goldberg 等人在 2012 年的文献 [167] 表明如果使用文献 [26] 提出的 *arc-standard* 算法，一个依存句法树可以与多种转移序列对应。

除了歧义问题，训练与测试不一致也是一个影响基于搜索的结构预测性能的问题。因为训练的目标是模仿参考策略，在模型学习过程中使用的训练状态都是正确的状态，这意味着所有状态都能够达到正确的结构。然而，在测试阶段，模型会犯错进入错误状态，因而无法达到正确的结构。由于训练过程没有学习如何在

表 5-1 采用基于搜索的结构预测建模基于转移的句法分析。

Table 5-1 The search-based structured prediction view of transition-based dependency parsing[26].

	Dependency parsing
s_t	(σ, β, A)，其中 σ 是一个栈，β 是一个缓存，A 代表部分建立的树
$\mathcal{A}$	{SHIFT, LEFT, RIGHT}
$\mathcal{S}_0$	$\{([\,], [1, .., n], \emptyset)\}$
$\mathcal{S}_T$	$\{([\text{ROOT}], [\,], A)\}$
$\mathcal{T}(s, a)$	• SHIFT: $(\sigma, j\|\beta) \rightarrow (\sigma\|j, \beta)$ • LEFT: $(\sigma\|i\ j, \beta) \rightarrow (\sigma\|j, \beta) \quad A \leftarrow A \cup \{i \leftarrow j\}$ • RIGHT: $(\sigma\|i\ j, \beta) \rightarrow (\sigma\|i, \beta) \quad A \leftarrow A \cup \{i \rightarrow j\}$

Input: training data: $\{\mathbf{x}^{(n)}, \mathbf{y}^{(n)}\}_{n=1}^{N}$; the reference policy: $\pi_{\mathcal{R}}(s, \mathbf{y})$.
Output: classifier $p(a|s)$.

$D \leftarrow \emptyset$;
for $n \leftarrow 1 \ldots N$ **do**
  $t \leftarrow 0$;
  $s_t \leftarrow s_0(\mathbf{x}^{(n)})$;
  while $s_t \notin \mathcal{S}_T$ **do**
    $a_t \leftarrow \pi_{\mathcal{R}}(s_t, \mathbf{y}^{(n)})$;
    $D \leftarrow D \cup \{s_t\}$;
    $s_{t+1} \leftarrow \mathcal{T}(s_t, a_t)$;
    $t \leftarrow t + 1$;
  end
end
optimize $\mathcal{L}_{NLL}$;

算法 5-1 基于搜索的结构预测的一般算法。

错误状态上做出决策，模型可能会给出不合理的预测。同时，由于贪心解码容易误差级联，这种错误可能会累积导致更大的错误。

5.2.2 知识蒸馏

由模型集成或大量参数组成的复杂模型往往具有较好的房型但运行速度较慢。知识蒸馏[161,176,177]是一类机器学习算法。这类算法可以将复杂模型（教师模型）的泛化能力转移到简单模型（学生模型）上。不同于传统的使用对数似然优化模型，知识蒸馏往往使用学生模型拟合教师模型的概率分布并尝试优化如下知识蒸馏学习目标

$$\mathcal{L}_{KD} = \sum_{x \in D} \sum_{y} -q(y \mid x) \cdot \log p(y \mid x)。 \tag{5-2}$$

在基于搜索的结构预测的情境下，x 与搜索状态 s 对应，y 与搜索动作 a 对应。教师模型的泛化能力可以通过优化知识蒸馏的学习目标 ‘‘蒸馏’’ 到学生模型上。当输入 x 的正确类别 y^* 已知时，知识蒸馏的学习目标可以与传统的对数似然学习目

标通过简单插值进行结合，即

$$\mathcal{L} = \alpha \mathcal{L}_{KD} + (1 - \alpha)\mathcal{L}_{NLL}。 \tag{5-3}$$

5.3 基于转移的依存句法分析中的知识蒸馏

5.3.1 模型集成

根据文献 [161]，尽管机器学习算法的 ‘‘最终目标’’ 是在新数据上取得好的效果，但实践中通常优化的是训练数据上的准确率。这种做法会使模型产生训练数据的偏置。在基于搜索的结构预测中，训练数据的歧义以及训练测试不一致往往会导致这种偏置。而对于噪声数据鲁棒性交差的学习目标往往会加剧这种偏置。

文献 [170] 研究了模型集成对于噪声数据的作用，并且经验性地证明集成学习能够克服训练数据的噪声。文献 [38] 通过将不同轮次获得的结构预测模型权重加权求和来近似集成学习的效果。参考上述工作，本章考虑使用集成学习提升模型的泛化性。在实践中，本章训练 M 个不同初始化的基于搜索的结构预测模型然后将其概率输出取平均 $q(a \mid s) = \frac{1}{M}\sum_m q_m(a \mid s)$ 作为模型集成。在第5.5.3.1节中，本章经验性的证明模型集成在有歧义的状态下表现更好。

5.3.2 从参考状态中蒸馏模型

如第5.5节所示，模型集成可以带来性能的提升。然而，模型的实际部署过程往往需要考虑计算与内存的开销。模型集成需要对一个实例预测多次，这限制了其在现实问题中的应用。为了能在只解码一次的情况下发挥集成学习的优势，本章考虑从集成模型中通过知识蒸馏学习到一个单模型。最简单的蒸馏方法是将算法5-1 中的对数似然学习目标转化为知识蒸馏学习目标（参考公式5-3）。算法5-2显示了这一方法。由于这种算法在参考状态上进行知识蒸馏学习，本章将其命名为*从参考状态中蒸馏模型*。图5-1中虚线连接的部分代表本章*从参考状态中蒸馏模型*的过程。

5.3.3 从探索状态中蒸馏模型

在基于搜索的结构预测的情景中，将教师模型的知识 ‘‘蒸馏’’ 到学生模型既包括在参考状态上模仿教师模型的概率分布，也包括模仿教师模型做出的决策。为了达到这一目标，本章提出使用教师模型随机采样出一系列的状态，并在这些状态上以知识蒸馏为目标学习学生模型。具体来讲，本章使用一个采样策略 $\pi_{\mathcal{E}}(s)$ 替代参考策略 $\pi_{\mathcal{R}}(s, \mathbf{y})$。$\pi_{\mathcal{E}}(s)$ 依照一个温度 T[①] 控制的函数 $q(a \mid s)^{\frac{1}{T}}$ 采样出 a。替换后的算法如算法5-2所示。由于这种算法在探索得到的状态上进行知识蒸馏学习，本

① 根据文献 [161]，温度控制采样函数的尖锐程度。

Input: training data: $\{\mathbf{x}^{(n)}, \mathbf{y}^{(n)}\}_{n=1}^{N}$; the reference policy: $\pi_{\mathcal{R}}(s, \mathbf{y})$; the exploration policy: $\pi_{\mathcal{E}}(s)$ which samples an action from the annealed ensemble $q(a \mid s)^{\frac{1}{T}}$
Output: classifier $p(a \mid s)$.

$D \leftarrow \emptyset$;
for $n \leftarrow 1...N$ **do**
  $t \leftarrow 0$;
  $s_t \leftarrow s_0(\mathbf{x}^{(n)})$;
  while $s_t \notin \mathcal{S}_T$ **do**
    if *distilling from reference* **then**
      $a_t \leftarrow \pi_{\mathcal{R}}(s_t, \mathbf{y}^{(n)})$;
    else
      $a_t \leftarrow \pi_{\mathcal{E}}(s_t)$;
    end
    $D \leftarrow D \cup \{s_t\}$;
    $s_{t+1} \leftarrow \mathcal{T}(s_t, a_t)$;
    $t \leftarrow t + 1$;
  end
end
if *distilling from reference* **then**
  optimize $\alpha\mathcal{L}_{KD} + (1 - \alpha)\mathcal{L}_{NLL}$;
else
  optimize $\mathcal{L}_{KD}$;
end

算法 5-2 基于搜索的结构预测的知识蒸馏算法。

章将其命名为从探索状态中蒸馏模型。图5-1中蓝色实线连接的部分代表本章从探索状态中蒸馏模型的过程。

由于参考策略 $\pi_{\mathcal{R}}$ 不是在采样状态上定义的，要在这类状态上使用对数似然学习往往比较困难。然而，在第5.5节，本章经验性地证明完全从知识蒸馏的目标中学习模型（即在公式5-3中设置 $\alpha = 1$）的可行性。

5.3.4 从两种状态中蒸馏模型

从参考状态中蒸馏模型鼓励模型按照参考动作进行预测；从探索状态中蒸馏模型使得模型能够在任意状态下进行学习。这两种学习方法从不同的角度将集成模型的泛化能力蒸馏到单模型中。两者可以进一步结合来达到更好的效果。本章按照下述方法从两种状态中进行知识蒸馏学习：首先使用 $\pi_{\mathcal{R}}$ 与 $\pi_{\mathcal{E}}$ 产生一系列的训练状态，然后在这些状态上学习 $p(a \mid s)$。如果一个状态是由 $\pi_{\mathcal{R}}$ 产生的，则最小化对数似然与知识蒸馏误差的插值；否则只最小化知识蒸馏的误差。算法5-2总结了整个学习过程。

5.4 基于上下文相关词向量的句法分析器的知识蒸馏

本章在基于转移的句法分析中验证本章的知识蒸馏方法。本章的参考文献 [7]

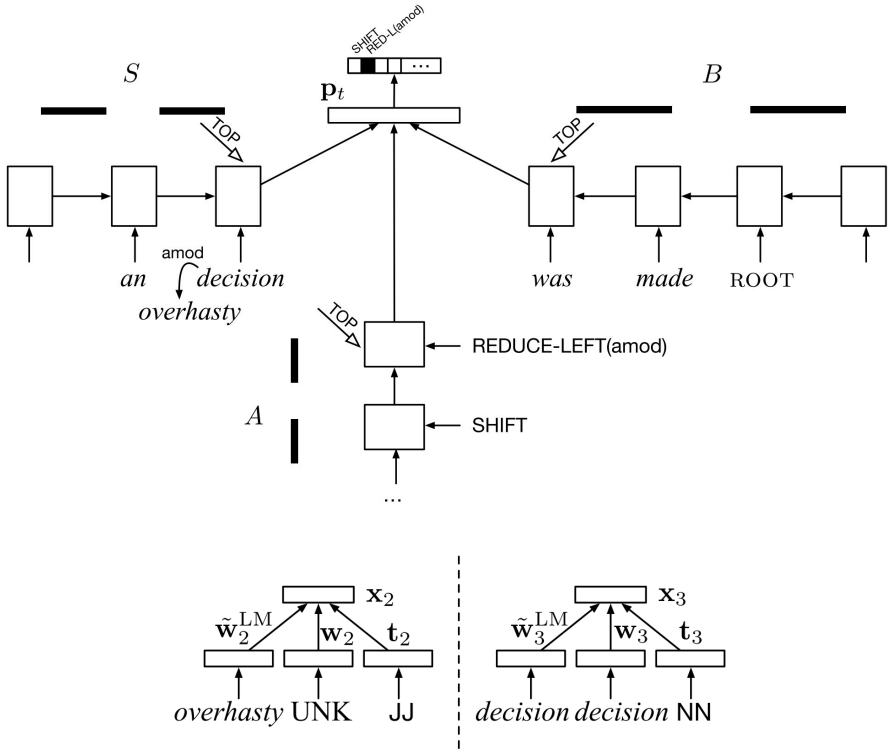

图 5-2 文献 [7] 提出的基于转移的依存句法分析器。
Fig. 5-2 The parser of Dyer et al. (2015) [7].

提出的 stack-lstm parser 构建本章的句法分析器。其框架如图5-2所示。对于 stack-lstm parser，其输入词向量包括固定的词向量 $\mathbf{p}$ 与可调的词向量 $\mathbf{w}$，其做法如图5-2所示。本章分别尝试对 stack-lstm parser 的集成模型以及基于上下文相关词向量的模型进行蒸馏。通过对基于上下文相关词向量的模型进行蒸馏，本章可以回答本章方法对教师模型与学生模型异构的情况下，能否通过知识蒸馏的方式达到又快又好的结构预测的效果？

本章在此使用与第4章类似的方式在基于转移的依存句法分析中加入上下文相关词向量。即将其各层取平均，作为一种额外的固定的向量表示。本章待蒸馏的模型则不含有对应的上下文相关词向量。在这一节，本章使用的蒸馏过程与前文从两种状态中蒸馏的方法相同。

5.5 实验

本章在基于转移的依存句法分析进行实验。本章依据第5.2.1节提到的方法将基于转移的句法分析转化为基于搜索的结构预测问题。

在基于转移的句法分析的实验中，本章使用 Dyer 等人在 2015 年的文献 [7] 中提出的 stack-lstm 算法建模策略的分类器。[①]

① 本章将句法分析代码开源在https://github.com/Oneplus/twpipe。

5.5.1 设置

本章在宾州树库（Penn Treebank，简称 PTB，文献 [65]）数据集上进行实验。本章使用标准数据划分（2-21 节用作训练集，22 节用作开发集，23 节用作测试集）。本章参考文献[7] 并使用斯坦福依存句法（Stanford dependencies）规范将短语结构句法转化为依存句法。① 本章通过 10-折交叉验证获得训练数据的自动词性，其准确率为 97.5%。本章使用不考虑标点符号的带标签依存关系准确率（LAS）评估通用依赖性分析性能。本章的模型的超参数参考文献 [7]。最好的迭代轮次由开发集性能决定。

文献 [84] 指出神经网络的学习过程是非确定的，所以汇报单次运行的结果并不能反映模型的性能。为了消除随机初始化对模型性能的影响，本章报告了 20 次随机运行的结果的平均值。

5.5.2 结果

表5-2显示了 PTB 数据集上的实验结果。这一结果显示集成模型高于基线模型 1.90。在将 α 设置为 1 时，模型取得了最佳的开发集结果。而此时测试集的准确率为 91.99。

本章也研究了温度对于从探索状态中蒸馏的模型的影响。图5-3显示了这一结果。根据表5-3，在采样时使用尖锐的分布总体上取得了更好的开发集性能。从探索状态蒸馏与从参考状态中蒸馏获得的模型的性能基本相当。将两者结合进一步提高了准确率（带标签的依存关系准确率达到 92.14）。

本章也在表5-2中将本章提出的模型与其他依存句法分析器进行了对比。表5-2的第二组显示了前人工作中基于贪心解码的依存句法分析器的性能。文献 [180] 使用了与本章不同的状态表示方法，同时分别研究了贪心解码与柱搜索（beam-search）的效果。文献 [179] 研究了使用 dynamic oracle 训练贪心解码的句法分析器的问题。本章提出的基于知识蒸馏的句法分析器的表现好于这些基于贪心解码的句法分析器。表5-2显示了其他依存句法分析算法的效果。这些算法包括：在基于转移的句法分析中使用柱搜索[180,181]，使用全局优化目标训练基于转移的依存句法分析[180]，基于图的依存句法分析[8]，将基于图的依存句法分析器蒸馏到基于转移的依存句法分析器[182]，以及将短语结构句法的结果通过规则转化为依存句法[183]。本章提出的基于知识蒸馏的方法由于这些基于转移的算法，但相较其他算法仍有差距。这类差距主要是由对问题的建模方式的不同导致的。

除了对集成模型进行蒸馏，本章也尝试对基于上下文相关词向量的句法分析

① 本章使用 Stanford CoreNLP 3.3.0 (https://stanfordnlp.github.io/CoreNLP/history.html) 完成这一转化。

表 5-2 句法分析的结果。显著性检验证明基于知识蒸馏的方法相对基线系统的提升以 $p \leq 0.01$ 统计显著。

Table 5-2 The dependency parsing results. Significance test[178] shows the improvement of our *Distill (both)* over *Baseline* is statistically significant with $p < 0.01$.

	LAS
Baseline	90.83
Ensemble (20)	92.73
Distill (reference, α=1.0)	91.99
Distill (exploration, T=1.0)	92.00
Distill (both)	92.14
Ballesteros 等人，文献 [179] (dyn. oracle)	91.42
Andor 等人，文献 [180] (local, B=1)	91.02
Buckman 等人，文献 [181] (local, B=8)	91.19
Andor 等人，文献 [180] (local, B=32)	91.70
Andor 等人，文献 [180] (global, B=32)	92.79
Dozat 与 Manning，文献 [8]	94.08
Kuncoro 等人，文献 [182]	92.06
Kuncoro 等人，文献 [183]	94.60

表 5-3 针对基于上下文相关词向量的句法分析的知识蒸馏实验。

Table 5-3 Distillation from structured prediction model with contextualized embeddings.

model	LAS	spd.
Baseline	90.83	1x
+ELMo	92.64	7.3x
Distill (both)	91.80	1x

器进行蒸馏。其实验结果如表5-3所示。根据表5-3，使用本章提出的知识蒸馏方法可以从异构模型中有效地学习模型。从而使用简单模型达到与使用上下文相关词向量模型类似的性能。由于本章简单模型不使用上下文相关词向量，所以，这一方法显著优化了模型速度。

5.5.3 分析

根据第5.5.2节的结果，使用集成模型显著提高了依存句法分析以及神经机器翻译的性能够。然而，类似 ‘‘为什么集成模型如此有效？能不能放弃传统对数似然学习目标，完全从知识蒸馏学习目标中学习？以及从知识蒸馏中学习是不是稳定的？’’ 一类的问题并没有得到很好的回答。在这一节中，本章首先研究集成模型在 ‘‘错误状态’’ 上的表现证明其具有更好的泛化能力。然后，本章通过研究公式5-3中的 a 的作用经验性地证明了完全从知识蒸馏中学习的可行性。最后，本章通过分析表明知识蒸馏可以带来更稳定的模型学习。

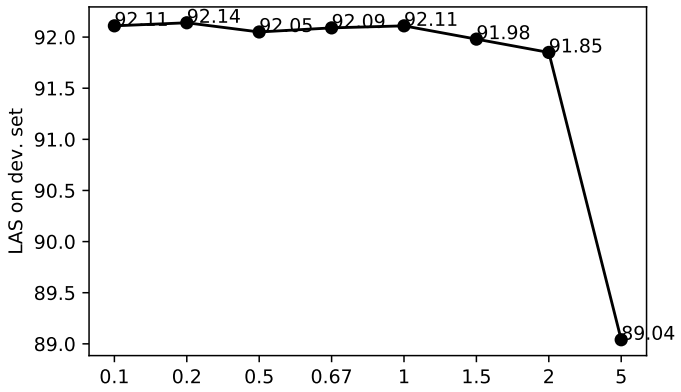

图 5-3 温度参数 T 对模型性能的影响。

Fig. 5-3 The effect of T on development set.

5.5.3.1 模型集成对于 ‘‘错误状态’’ 的作用

如前文所述，歧义或者非最优的 ‘‘错误状态’’ 影响结构预测的准确率。根据第5.5.2节的结果，模型集成能够提升性能，这表明其能够更好地处理 ‘‘错误状态’’。为了经验性地验证这一假设，本章以依存句法分析为目标，借助 dynamic oracle 研究集成模型的输出分布。

dynamic oracle[167,184] 能够有效地判断，对于一棵句法树的任意给定状态 s，最优的转移动作（*参考动作*）是什么。这里 ‘‘最优’’ 指的是能够搜索得到的相似度（用 LAS 衡量）最高的树。这种性质使得本章可以对 ‘‘错误状态’’ 中的某个动作进行分析。本章根据*参考动作*评价了基线模型以及集成模型在动作空间上输出的分布。由于 dynamic oracle 会对一个状态给出多个参考动作，本章将这一评价问题转化为一个排序的评价问题。直观来讲，当多个参考动作存在是，一个好的句法分析器应当将动作的概率密度集中在参考动作上。本章通过随机采样从基线模型中获得了一系列状态。不同模型的排序分数如表5-4所示。根据表5-4，集成模型在 ‘‘错误状态’’ 下的排序表现更好。这一观察表明，集成模型的输出分布所包含的信息更丰富，因而在 ‘‘错误状态’’ 上表现得更好，并取得更好的性能。同时，本章也观察到蒸馏模型的性能好于基线与集成模型。本章认为造成这一现象的原因是蒸馏模型的学习过程包含探索机制。

5.5.3.2 α 的效果

本章在从参考状态中蒸馏模型的学习过程中研究 α 的影响。本章在基于转移的句法分析以及神经机器翻译中将 α 从 0 到 1 以 0.1 为步长变化，并在图5-4中展示了模型的性能。两个实验都显示大 α 的表现好于小 α。对于句法分析，$\alpha = 1$ 时取得了最好的开发集性能。对于神经机器翻译，开发集性能最好的 α 是 0.8。同时，在机器翻译实验中，$\alpha = 0.8$ 与 $\alpha = 1$ 的模型只有 0.2 的差距。这一系列观察表明

表 5-4 不同模型的 ‘‘错误状态’’ 的排序性能。

Table 5-4 The ranking performance of parsers' output distributions evaluated in MAP on "problematic" states.

	有歧义的正确状态	非正确状态
Baseline	68.59	89.59
Ensemble	74.19	90.90
Distill (both)	81.15	91.38

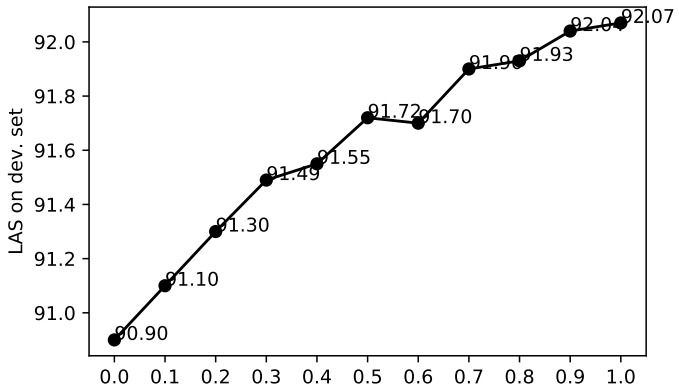

图 5-4 开发集上 α 值的效果

Fig. 5-4 The effect of α

对于从参考状态时，更多地关注知识蒸馏的学习目标能够带来更好的模型性。这一观察也显示对于从探索状态中蒸馏模型（即 $\alpha = 1$）的可行性。

5.5.3.3 模型学习的稳定性

除了提高性能，知识蒸馏也能给模型学习的稳定性带来提升。本章将不同随机种子得到的基线模型与蒸馏模型的准确率以琴图的方式绘制在图5-5中，同时在表5-5中显示了两种模型的标准差。可见，知识蒸馏模型的标准差更小，即其学习更稳定。文献 [185] 指出影响神经网络模型的泛化性能的主要问题并非过拟合，而是模型参数进入尖锐的局部极小点（sharp minimizer）。这种局部极小点的泛化性往往较差。根据这一理论，本章猜测知识蒸馏取得更稳定的训练效果的原因是知识蒸馏的学习曲目更光滑，尖锐的局部极小点更少。

5.6 相关工作

前人工作中有若干在 NLP 任务中使用知识蒸馏技术。Kim 与 Rush 在 2016 年的文献 [9] 中关注如将复杂模型蒸馏到简单模型中。同时，他们的方法主要关注序列级别的蒸馏。与之相反的是，本章更关注动作级的知识蒸馏，同时提出从参考状态以及从探索状态中进行知识蒸馏。

Freitag 等人在 2017 年的文献 [186] 中使用六个翻译器的译文生成翻译的训练

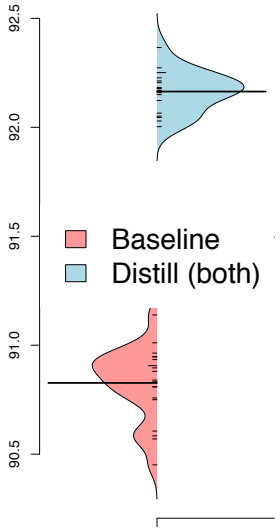

图 5-5 不同随机种子下模型分数的分布。

Fig. 5-5 The distributions of scores for the baseline model and our *distillation from both* on PTB on differently-seeded runs.

表 5-5 结果的最大值、最小值以及标准差。

Table 5-5 The minimal, maximum, and standard derivation values on differently-seeded runs.

system	seeds	min	max	σ
PTB test				
Baseline	20	90.45	91.14	0.17
Distill (both)	20	92.00	92.37	0.09

数据，其模型使用柱搜索进行探索。与其不同的是，本章使用单模型产生训练数据，并以拟合集成模型的分布为学习目标。

在强化学习中，集成模型的探索机制也得到了相应的研究[187]。除了在标注数据上蒸馏模型，也有一部分半监督学习的工作从生语料上进行模型学习[92,188]。如何在生语料上进行知识蒸馏也值得后续工作的研究。

Kuncoro 在 2016 年的文献 [182] 中提出将 20 个基于转移的句法分析器的结果通过投票的方式进行集成并将结果以后验正则的方式应用于训练一个基于图的模型。不同于他们的模型，将基于转移的模型蒸馏到基于转移的模型中。

除了使用知识蒸馏，Stahlberg 等人在 2017 年的文献 [189] 中提出将若干翻译模型的集成组合为一个神经网络，并用 SVD 化简了模型。他们的模型也可以视作一种知识蒸馏。

5.7 本章小结

本章研究了结构预测中的知识蒸馏问题。本章提出将若干模型的模型集成蒸

馏到一个单模型中。这一蒸馏过程同时在参考状态以及探索状态中完成。在基于转移的句法分析实验中，本章模型获得了显著的性能提升，从而证明了本章提出的方法的有效性。同时，本章通过分析给出了本章知识蒸馏方法可行性的验证。

在此基础上，本章对于基于上下文相关词向量的句法分析器进行知识蒸馏。实验结果显示本章提出的方法对于异构模型同样有效，在少量损失精度的情况下近十倍地提高模型速度。

结 论

包括分词、词性标注、命名实体识别以及句法分析在内的语言分析是基础而重要的自然语言处理问题。提高语言分析的性能能够有效地帮助下游任务。强化词表示能力是提高语言分析性能的一种有效手段。近年来提出的上下文相关词向量已经帮助一部分自然语言处理任务取得了性能提升。本文受到相关研究的启发，针对上下文相关词向量与语言分析的结合开展了一系列研究。

本文针对现有上下文相关词向量因过分强调使用多层网络对一整句甚至多句进行抽象而导致的效率问题，从语言分析主要依赖局部信息出发，提出一种融合相对位置权重的窗口级自注意力机制并将其应用于上下文表示，从而获得一种适用于语言分析的上下文相关词向量。本文在五项语言分析任务上进行实验，相应结果表明本文提出的上下文相关词向量在不损失精度的情况下获得了三倍的速度提升。

基于前文提出的上下文相关词向量，本文针对语言分析中的切分问题（中文分词与命名实体识别）对合理的片段（词与实体）表示的依赖，本文在上下文相关词向量的基础上提出一种基于简单拼接的片段表示方法并将其应用于半-马尔科夫条件随机场中。在典型切分问题的实验中，本文基于上下文相关词向量的片段表示有效地提高了模型性能。通过进一步融合任务相关的上下文表示以及建模片段级信息的片段向量，本文模型取得了当前最优的性能。

基于前文提出的上下文相关词向量，本文针对上下文相关词向量对多国语句法分析作用尚无明确结论的现状，提出在多国语句法分析中使用上下文相关词向量并在大规模树库上验证上下文相关词向量的有效性。除了获得稳定且显著的性能提升，本文针对提升的原因进行了详细的分析。大量分析实验表明，性能提升的主要原因是上下文相关词向量通过对于未登录词词形的更好的建模有效地提升了未登录词的准确率。

针对使用上下文相关词向量的句法分析参数过多、运行速度较慢的问题，本文提出一种结合探索机制的知识蒸馏算法，以将基于上下文相关词向量的复杂模型用不使用相应词向量的简单模型进行近似，从而在不显著降低性能的情况下提高句法分析速度。实验结果表明，本文提出的方法在损失少量句法分析准确率的情况下，近十倍地提升了速度。

本文研究表明上下文相关词向量可以很好地与语法分析结合，从而显著提高

语法分析的性能。同时，本文也以语言分析为工具对于上下文相关词向量进行全面的分析。这些分析表明上下文相关词向量具有表示未登录词，建模片段的能力。

参考文献

[1] Mikolov T, Chen K, Corrado G, et al. Efficient Estimation of Word Representations in Vector Space[C/OL] // 1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proc.. 2013. http://arxiv.org/abs/1301.3781.

[2] Mikolov T, Sutskever I, Chen K, et al. Distributed Representations of Words and Phrases and their Compositionality[G/OL] // NIPS 26. 2013 : 3111-3119. http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf.

[3] Pennington J, Socher R, Manning C. Glove: Global Vectors for Word Representation[C] // Proc. of EMNLP. 2014.

[4] Bojanowski P, Grave E, Joulin A, et al. Enriching Word Vectors with Subword Information[J/OL]. Transactions of the Association for Computational Linguistics, 2017, 5(1) : 135-146. https://www.aclweb.org/anthology/Q17-1010.

[5] Huang Z, Xu W, Yu K. Bidirectional LSTM-CRF Models for Sequence Tagging[J]. CoRR, 2015, abs/1508.01991.

[6] Lample G, Ballesteros M, Subramanian S, et al. Neural Architectures for Named Entity Recognition[C] // Proc. of NAACL. 2016.

[7] Dyer C, Ballesteros M, Ling W, et al. Transition-Based Dependency Parsing with Stack Long Short-Term Memory[C] // Proc. of ACL. 2015.

[8] Dozat T, Manning C D. Deep Biaffine Attention for Neural Dependency Parsing[J]. CoRR, 2016, abs/1611.01734.

[9] Kim Y, Rush A M. Sequence-Level Knowledge Distillation[C] // Proc. of EMNLP. 2016.

[10] Firth J R. A synopsis of linguistic theory 1930-55.[J], , 1952-59 : 1-32.

[11] Guo J, Che W, Wang H, et al. Learning Sense-specific Word Embeddings By Exploiting Bilingual Resources[C] // Proc. of Coling. 2014.

[12] McCann B, Bradbury J, Xiong C, et al. Learned in Translation: Contextualized Word Vectors[G] // NIPS 30. 2017 : 6294-6305.

[13] Peters M, Neumann M, Iyyer M, et al. Deep Contextualized Word Representations[C] // Proc. of NAACL. 2018.

[14] Howard J, Ruder S. Universal Language Model Fine-tuning for Text Classification[C] // Proc. of ACL. 2018.

[15] Radford A, Narasimhan K, Salimans T, et al. Improving Language Understanding by Generative Pre-Training[H]. 2018.

[16] Devlin J, Chang M, Lee K, et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding[J]. CoRR, 2018, abs/1810.04805.

[17] Hochreiter S, Schmidhuber J. Long Short-Term Memory[J/OL]. Neural Comput., 1997, 9(8). http://dx.doi.org/10.1162/neco.1997.9.8.1735.

[18] Vaswani A, Shazeer N, Parmar N, et al. Attention is All you Need[G] // NIPS 30. 2017 : 5998-6008.

[19] Xue N, Shen L. Chinese Word Segmentation As LMR Tagging[C/OL] // Proc. of the Second SIGHAN Workshop on Chinese Language Processing - Volume 17. 2003. http://dx.doi.org/10.3115/1119250.1119278.

[20] Zhang Y, Clark S. Chinese Segmentation with a Word-Based Perceptron Algorithm[C] // Proc. of ACL. 2007.

[21] Zheng X, Chen H, Xu T. Deep Learning for Chinese Word Segmentation and POS Tagging[C] // Proc. of EMNLP. 2013.

[22] Toutanova K, Klein D, Manning C D, et al. Feature-rich Part-of-speech Tagging with a Cyclic Dependency Network[C/OL] // Proc. of the NAACL '03. 2003. http://dx.doi.org/10.3115/1073445.1073478.

[23] Ma X, Hovy E. End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF[C] // Proc. of ACL. 2016.

[24] Ando R K, Zhang T. A Framework for Learning Predictive Structures from Multiple Tasks and Unlabeled Data[J]. J. Mach. Learn. Res., 2005, 6.

[25] McDonald R T, Pereira F C. Online Learning of Approximate Dependency Parsing Algorithms.[C] // Proc. of EACL. 2006.

[26] Nivre J. Algorithms for deterministic incremental dependency parsing[J]. Computational Linguistics, 2008, 34(4).

[27] Zhang Y, Clark S. Joint Word Segmentation and POS Tagging Using a Single Perceptron[C] // Proc. of ACL. 2008.

[28] Chen D, Manning C. A Fast and Accurate Dependency Parser using Neural Networks[C] // Proc. of EMNLP. 2014.

[29] Xue N, Palmer M. Calibrating Features for Semantic Role Labeling[C/OL] // Proc. of EMNLP 2004. 2004. https://www.aclweb.org/anthology/W04-3212.

[30] Li J, Zhou G, Ng H T. Joint Syntactic and Semantic Parsing of Chinese[C/OL] // Proc. of ACL. 2010. https://www.aclweb.org/anthology/P10-1113.

[31] Dyer C. Using a maximum entropy model to build segmentation lattices for MT[C/OL] // Proc. of NAACL. 2009. https://www.aclweb.org/anthology/N09-1046.

[32] Aharoni R, Goldberg Y. Towards String-To-Tree Neural Machine Translation[C/OL] // Proc. of ACL. 2017. https://www.aclweb.org/anthology/P17-2021.

[33] Brown P F, deSouza P V, Mercer R L, et al. Class-based N-gram Models of Natural Language[J]. Comput. Linguist., 1992, 18(4).

[34] Brown P F, Pietra V J D, Pietra S A D, et al. The Mathematics of Statistical Machine Translation: Parameter Estimation[J/OL]. Comput. Linguist., 1993, 19(2) : 263-311. http://dl.acm.org/citation.cfm?id=972470.972474.

[35] Vogel S, Ney H, Tillmann C. HMM-based Word Alignment in Statistical Translation[C/OL] // Proc. of the 16th Conference on Computational Linguistics - Volume 2. 1996. https://doi.org/10.3115/993268.993313.

[36] Eisner J. Three New Probabilistic Models for Dependency Parsing: An Exploration[C] // Proc. of Coling. 1996.

[37] Collins M. Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms[C/OL] // Proc. of EMNLP. 2002. http://dx.doi.org/10.3115/1118693.1118694.

[38] Daumé III H, Langford J, Marcu D. Search-Based Structured Prediction as Classification[C] // NIPS Workshop on ASLTSP. 2005.

[39] Zhang Y, Clark S. Syntactic Processing Using the Generalized Perceptron and Beam Search[J]. Computational Linguistics, 2011, 37(1).

[40] Taskar B, Guestrin C, Koller D. Max-Margin Markov Networks[G/OL] // NIPS 16. 2004 : 25-32. http://papers.nips.cc/paper/2397-max-margin-markov-networks.pdf.

[41] Crammer K, Dekel O, Keshet J, et al. Online Passive-Aggressive Algorithms[J]. J. Mach. Learn. Res., 2006, 7.

[42] Hubel D H, Wiesel T N. Receptive Fields and Functional Architecture of Monkey Striate Cortex[J]. Journal of Physiology (London), 1968, 195 : 215-243.

[43] Collobert R, Weston J, Bottou L, et al. Natural Language Processing (Almost) from Scratch[J]. J. Mach. Learn. Res., 2011.

[44] Faruqui M, Tsvetkov Y, Rastogi P, et al. Problems With Evaluation of Word Embeddings Using Word Similarity Tasks[C] // Proc. of the 1st Workshop on Evaluating Vector-Space Representations for NLP. 2016.

[45] Lai S, Liu K, He S, et al. How to Generate a Good Word Embedding[J/OL]. IEEE Intelligent Systems, 2016, 31(6) : 5-14. http://dx.doi.org/10.1109/MIS.2016.45.

[46] Schnabel T, Labutov I, Mimno D, et al. Evaluation methods for unsupervised word embeddings[C/OL] // Proc. of EMNLP. 2015. http://aclweb.org/anthology/D15-1036.

[47] Lei T, Xin Y, Zhang Y, et al. Low-Rank Tensors for Scoring Dependency Structures[C/OL] // Proc. of ACL. 2014. http://www.aclweb.org/anthology/P14-1130.

[48] Lei T, Zhang Y, Màrquez L, et al. High-Order Low-Rank Tensors for Semantic Role Labeling[C/OL] // Proc. of NAACL. 2015. http://www.aclweb.org/anthology/N15-1121.

[49] Lund K, Burgess C. Producing high-dimensional semantic spaces from lexical co-occurrence[J]. Behavior research methods, instruments, & computers, 1996, 28(2) : 203-208.

[50] Rohde D L T, Gonnerman L M, Plaut D C. An improved model of semantic similarity based on lexical co-occurence[J]. COMMUNICATIONS OF THE ACM, 2006, 8 : 627-633.

[51] Lebret R, Collobert R. Word Embeddings through Hellinger PCA[C] // Proc. of EACL. 2014.

[52] Levy O, Goldberg Y. Neural Word Embedding as Implicit Matrix Factorization[G] // NIPS 27. 2014 : 2177-2185.

[53] Bengio Y, Ducharme R, Vincent P, et al. A Neural Probabilistic Language Model[J]. J. Mach. Learn. Res., 2003, 3 : 1137-1155.

[54] Bengio Y, Ducharme R, Vincent P. A Neural Probabilistic Language Model[G] // NIPS 13. 2001 : 932-938.

[55] Mnih A, Hinton G. Three New Graphical Models for Statistical Language Modelling[C/OL] // Proc. of ICML. 2007. http://dx.doi.org/10.1145/1273496.1273577.

[56] Mnih A, Hinton G E. A Scalable Hierarchical Distributed Language Model[G] // NIPS 21. 2009 : 1081-1088.

[57] Mikolov T, Karafiát M, Burget L, et al. Recurrent neural network based language model[C] // INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010. 2010.

[58] Ling W, Dyer C, Black A W, et al. Two/Too Simple Adaptations of Word2Vec for Syntax Problems[C] // Proc. of NAACL. 2015.

[59] Rothe S, Schütze H. AutoExtend: Extending Word Embeddings to Embeddings for Synsets and Lexemes[C] // Proc. of ACL. 2015.

[60] Smith S L, Turban D H P, Hamblin S, et al. Offline bilingual word vectors, orthogonal transformations and the inverted softmax[J]. CoRR, 2017, abs/1702.03859.

[61] Guo J, Che W, Yarowsky D, et al. A Representation Learning Framework for Multi-Source Transfer Parsing[C/OL] // . 2016. https://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12236.

[62] Tang D, Wei F, Yang N, et al. Learning Sentiment-Specific Word Embedding for Twitter Sentiment Classification[C/OL] // Proc. of ACL. 2014. http://www.aclweb.org/anthology/P14-1146.

[63] Yin W, Schütze H. Learning Word Meta-Embeddings[C] // Proc. of ACL. 2016.

[64] Tjong Kim Sang E F, De Meulder F. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition[C] // Proc. of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003. 2003.

[65] Marcus M P, Santorini B, Marcinkiewicz M A. Building a Large Annotated Corpus of English: The Penn Treebank[J]. Computational Linguistic, 1993, 19(2) : 313-330.

[66] Melamud O, Goldberger J, Dagan I. context2vec: Learning Generic Context Embedding with Bidirectional LSTM[C] // Proc. of CoNLL. 2016.

[67] Sutskever I, Vinyals O, Le Q V. Sequence to Sequence Learning with Neural Networks[G] // NIPS 27. 2014 : 3104-3112.

[68] Luong T, Pham H, Manning C D. Effective Approaches to Attention-based Neural Machine Translation[C] // Proc. of EMNLP. 2015.

[69] Bahdanau D, Cho K, Bengio Y. Neural Machine Translation by Jointly Learning to Align and Translate[J/OL]. CoRR, 2014, abs/1409.0473. http://arxiv.org/abs/1409.0473.

[70] Socher R, Bauer J, Manning C D, et al. Parsing with Compositional Vector Grammars[C/OL] // Proc. of ACL. 2013. http://www.aclweb.org/anthology/P13-1045.

[71] Voorhees E M. The TREC-8 Question Answering Track Report[C/OL] // Proc. of The Eighth Text REtrieval Conference, TREC 1999, Gaithersburg, Maryland, USA, November 17-19, 1999. 1999. http://trec.nist.gov/pubs/trec8/papers/qa_report.pdf.

[72] Bowman S R, Angeli G, Potts C, et al. A large annotated corpus for learning natural language inference[C/OL] // Proc. of EMNLP. 2015. http://aclweb.org/anthology/D15-1075.

[73] Rajpurkar P, Zhang J, Lopyrev K, et al. SQuAD: 100,000+ Questions for Machine Comprehension of Text[C/OL] // Proc. of EMNLP. 2016. https://aclweb.org/anthology/D16-1264.

[74] He K, Zhang X, Ren S, et al. Deep Residual Learning for Image Recognition[J/OL]. CoRR, 2015, abs/1512.03385. http://arxiv.org/abs/1512.03385.

[75] Pradhan S, Moschitti A, Xue N, et al. Towards Robust Linguistic Analysis using OntoNotes[C/OL] // Proc. of CoNLL. 2013. https://www.aclweb.org/anthology/W13-3516.

[76] Pradhan S, Moschitti A, Xue N, et al. CoNLL-2012 Shared Task: Modeling Multilingual Unrestricted Coreference in OntoNotes[C/OL] // Joint Conference on EMNLP and CoNLL - Shared Task. 2012. https://www.aclweb.org/anthology/W12-4501.

[77] Peters M, Neumann M, Zettlemoyer L, et al. Dissecting Contextual Word Embeddings: Architecture and Representation[C] // Proc. of EMNLP. 2018.

[78] Dauphin Y N, Fan A, Auli M, et al. Language Modeling with Gated Convolutional Networks[C] // Proc. of ICML : Vol 70. 2017.

[79] Kim Y, Jernite Y, Sontag D, et al. Character-aware Neural Language Models[C/OL] // Proc. of the Thirtieth AAAI Conference on Artificial Intelligence. 2016. http://dl.acm.org/citation.cfm?id=3016100.3016285.

[80] Jean S, Cho K, Memisevic R, et al. On Using Very Large Target Vocabulary for Neural Machine Translation[C] // Proc. of ACL. 2015.

[81] Guo J, Che W, Wang H, et al. A Universal Framework for Inductive Transfer Parsing across Multi-typed Treebanks[C] // Proc. of COLING. 2016.

[82] Clark K, Luong M-T, Manning C D, et al. Semi-Supervised Sequence Modeling with Cross-View Training[C/OL] // Proc. of EMNLP. 2018. http://www.aclweb.org/anthology/D18-1217.

[83] Peters M, Ammar W, Bhagavatula C, et al. Semi-supervised sequence tagging with bidirectional language models[C/OL] // Proc. of ACL. 2017. http://aclweb.org/anthology/P17-1161.

[84] Reimers N, Gurevych I. Reporting Score Distributions Makes a Difference: Performance Study of LSTM-networks for Sequence Tagging[C] // Proc. of EMNLP. 2017.

[85] Cai D, Zhao H. Neural Word Segmentation Learning for Chinese[C/OL] // Proc. of ACL. 2016. http://www.aclweb.org/anthology/P16-1039.

[86] Zhou H, Yu Z, Zhang Y, et al. Word-Context Character Embeddings for Chinese Word Segmentation[C/OL] // Proc. of EMNLP. 2017. https://www.aclweb.org/anthology/D17-1079.

[87] Weiss D, Alberti C, Collins M, et al. Structured Training for Neural Network Transition-Based Parsing[C] // Proc. of ACL. 2015.

[88] Kiperwasser E, Goldberg Y. Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations[J]. TACL, 2016, 4.

[89] Dozat T, Qi P, Manning C D. Stanford's Graph-based Neural Dependency Parser at the CoNLL 2017 Shared Task[C] // Proc. of CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies. 2017.

[90] McCallum A, Freitag D, Pereira F C N. Maximum Entropy Markov Models for Information Extraction and Segmentation[C] // ICML. 2000.

[91] Lafferty J D, McCallum A, Pereira F C N. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data[C] // ICML 01. 2001.

[92] Liang P, Daumé H, Klein D. Structure compilation: trading structure for features[J], 2008 : 592-599.

[93] Ma J, Ganchev K, Weiss D. State-of-the-art Chinese Word Segmentation with Bi-LSTMs[C/OL] // Proc. of EMNLP. 2018. http://www.aclweb.org/anthology/D18-1529.

[94] Lee K, He L, Zettlemoyer L. Higher-Order Coreference Resolution with Coarse-to-Fine Inference[C/OL] // Proc. of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers). 2018. https://www.aclweb.org/anthology/N18-2108.

[95] Kitaev N, Klein D. Constituency Parsing with a Self-Attentive Encoder[C/OL] // Proc. of ACL. 2018. http://www.aclweb.org/anthology/P18-1249.

[96] Joshi V, Peters M, Hopkins M. Extending a Parser to Distant Domains Using a Few Dozen Partially Annotated Examples[C] // Proc. of ACL. 2018.

[97] Chelba C, Mikolov T, Schuster M, et al. One billion word benchmark for measuring progress in statistical language modeling[C/OL] // INTERSPEECH 2014, 15th Annual Conference of the International Speech Communication Association, Singapore, September 14-18, 2014. 2014. http://www.isca-speech.org/archive/interspeech_2014/i14_2635.html.

[98] Strubell E, Verga P, Belanger D, et al. Fast and Accurate Entity Recognition with Iterated Dilated Convolutions[C/OL] // Proc. of EMNLP. 2017. https://www.aclweb.org/anthology/D17-1283.

[99] Li L H, Chen P H, Hsieh C-J, et al. Efficient Contextual Representation Learning Without Softmax Layer[H]. 2019.

[100] Luong T, Socher R, Manning C. Better Word Representations with Recursive Neural Networks for Morphology[C/OL] // Proc. of CoNLL. 2013. http://www.aclweb.org/anthology/W13-3512.

[101] Botha J A, Blunsom P. Compositional Morphology for Word Representations and Language Modelling[C/OL] // Proc. of ICML. 2014. http://dl.acm.org/citation.cfm?id=3044805.3045104.

[102] Shaw P, Uszkoreit J, Vaswani A. Self-Attention with Relative Position Representations[C/OL] // Proc. of NAACL. 2018. https://www.aclweb.org/anthology/N18-2074.

[103] Srivastava R K, Greff K, Schmidhuber J. Highway Networks[J/OL]. CoRR, 2015, abs/1505.00387. http://arxiv.org/abs/1505.00387.

[104] Kingma D P, Ba J. Adam: A Method for Stochastic Optimization[J]. CoRR, 2014, abs/1412.6980.

[105] Tjong Kim Sang E F, Buchholz S. Introduction to the CoNLL-2000 Shared Task: Chunking[C/OL] // Proc. of CoNLL. 2000. https://doi.org/10.3115/1117601.1117631.

[106] Nivre J, de Marneffe M-C, Ginter F, et al. Universal Dependencies v1: A Multilingual Treebank Collection[C] // Proc. of LREC-2016. 2016.

[107] Gal Y, Ghahramani Z. A Theoretically Grounded Application of Dropout in Recurrent Neural Networks[C/OL] // Proc. of the 30th International Conference on Neural Information Processing Systems. 2016. http://dl.acm.org/citation.cfm?id=3157096.3157211.

[108] Elman J L. Finding structure in time[J]. COGNITIVE SCIENCE, 1990, 14(2): 179-211.

[109] Akbik A, Blythe D, Vollgraf R. Contextual String Embeddings for Sequence Labeling[C/OL] // Proc. of Coling. 2018. http://www.aclweb.org/anthology/C18-1139.

[110] Yasunaga M, Kasai J, Radev D. Robust Multilingual Part-of-Speech Tagging via Adversarial Training[C/OL] // Proc. of NAACL. 2018. https://www.aclweb.org/anthology/N18-1089.

[111] Zhang M, Zhang Y, Fu G. End-to-End Neural Relation Extraction with Global Optimization[C/OL] // Proc. of EMNLP. 2017. https://www.aclweb.org/anthology/D17-1182.

[112] He L, Lee K, Lewis M, et al. Deep Semantic Role Labeling: What Works and What's Next[C] // Proc. of ACL. 2017.

[113] Gardner M, Grus J, Neumann M, et al. AllenNLP: A Deep Semantic Natural Language Processing Platform[C/OL] // Proc. of Workshop for NLP Open Source Software (NLP-OSS). 2018. http://aclweb.org/anthology/W18-2501.

[114] Lee K, He L, Zettlemoyer L. Higher-Order Coreference Resolution with Coarse-to-Fine Inference[C] // Proc. of NAACL. 2018.

[115] Okanohara D, Miyao Y, Tsuruoka Y, et al. Improving the Scalability of Semi-Markov Conditional Random Fields for Named Entity Recognition[C/OL] // Proc. of ACL. 2006. http://dx.doi.org/10.3115/1220175.1220234.

[116] Yang B, Cardie C. Extracting Opinion Expressions with semi-Markov Conditional Random Fields[C] // Proc. of EMNLP. 2012.

[117] Andrew G. A Hybrid Markov/Semi-Markov Conditional Random Field for Sequence Segmentation[C] // Proc. of EMNLP. 2006.

[118] Sarawagi S, Cohen W W. Semi-Markov Conditional Random Fields for Information Extraction[G] // NIPS 17. 2004.

[119] Kong L, Dyer C, Smith N A. Segmental Recurrent Neural Networks[J]. CoRR, 2015, abs/1511.06018.

[120] Zhuo J, Cao Y, Zhu J, et al. Segment-Level Sequence Modeling using Gated Recursive Semi-Markov Conditional Random Fields[C/OL] // Proc. of ACL. 2016. http://www.aclweb.org/anthology/P16-1134.

[121] Kalchbrenner N, Grefenstette E, Blunsom P. A Convolutional Neural Network for Modelling Sentences[C/OL] // Proc. of ACL. 2014. http://www.aclweb.org/anthology/P14-1062.

[122] Wang W, Chang B. Graph-based Dependency Parsing with Bidirectional LSTM[C/OL] // Proc. of ACL. 2016. http://www.aclweb.org/anthology/P16-1218.

[123] Cross J, Huang L. Span-Based Constituency Parsing with a Structure-Label System and Provably Optimal Dynamic Oracles[C/OL] // Proc. of EMNLP. 2016. https://aclweb.org/anthology/D16-1001.

[124] Chen W, Kazama J, Uchimoto K, et al. Improving Dependency Parsing with Subtrees from Auto-Parsed Data[C] // Proc. of EMNLP. 2009.

[125] Wang Y, Kazama J, Tsuruoka Y, et al. Improving Chinese Word Segmentation and POS Tagging with Semi-supervised Methods Using Large Auto-Analyzed Data[C] // IJCNLP-2011. 2011.

[126] Pei W, Ge T, Chang B. Max-Margin Tensor Neural Network for Chinese Word Segmentation[C] // Proc. of ACL. 2014.

[127] Guo J, Che W, Wang H, et al. Revisiting Embedding Features for Simple Semi-supervised Learning[C] // Proc. of EMNLP. 2014.

[128] Jiang W, Sun M, Lü Y, et al. Discriminative Learning with Natural Annotations: Word Segmentation as a Case Study[C] // Proc. of ACL. 2013.

[129] Pei W, Ge T, Chang B. An Effective Neural Network Model for Graph-based Dependency Parsing[C/OL] // Proc. of ACL. 2015. http://www.aclweb.org/anthology/P15-1031.

[130] Maaten L v d, Hinton G. Visualizing data using t-SNE[J]. Journal of machine learning research, 2008, 9.

[131] Liu L, Shang J, Ren X, et al. Empower Sequence Labeling with Task-Aware Neural Language Model[C] // AAAI. 2018.

[132] Hashimoto K, xiong c, Tsuruoka Y, et al. A Joint Many-Task Model: Growing a Neural Network for Multiple NLP Tasks[C/OL] // Proc. of EMNLP. 2017. https://www.aclweb.org/anthology/D17-1206.

[133] Søgaard A, Goldberg Y. Deep multi-task learning with low level tasks supervised at lower layers[C/OL] // Proc. of ACL. 2016. http://anthology.aclweb.org/P16-2038.

[134] Suzuki J, Isozaki H. Semi-Supervised Sequential Labeling and Segmentation Using Giga-Word Scale Unlabeled Data[C] // Proc. of ACL. 2008.

[135] Yang J, Zhang Y. NCRF++: An Open-source Neural Sequence Labeling Toolkit[C/OL] // Proc. of ACL 2018, System Demonstrations. 2018. http://www.aclweb.org/anthology/P18-4013.

[136] Ye Z, Ling Z-H. Hybrid semi-Markov CRF for Neural Sequence Labeling[C/OL] // Proc. of ACL. 2018. http://www.aclweb.org/anthology/P18-2038.

[137] Peng N, Dredze M. Improving Named Entity Recognition for Chinese Social Media with Word Segmentation Representation Learning[C/OL] // Proc. of ACL. 2016. http://anthology.aclweb.org/P16-2025.

[138] Yang J, Zhang Y, Dong F. Neural Word Segmentation with Rich Pretraining[C/OL] // Proc. of ACL. 2017. http://aclweb.org/anthology/P17-1078.

[139] Chen X, Shi Z, Qiu X, et al. Adversarial Multi-Criteria Learning for Chinese Word Segmentation[C/OL] // Proc. of ACL. 2017. http://aclweb.org/anthology/P17-1110.

[140] Cai D, Zhao H, Zhang Z, et al. Fast and Accurate Neural Word Segmentation for Chinese[C/OL] // Proc. of ACL. 2017. http://aclweb.org/anthology/P17-2096.

[141] Chen X, Qiu X, Zhu C, et al. Gated Recursive Neural Network for Chinese Word Segmentation[C/OL] // Proc. of ACL. 2015. http://www.aclweb.org/anthology/P15-1168.

[142] Chen X, Qiu X, Zhu C, et al. Long Short-Term Memory Neural Networks for Chinese Word Segmentation[C/OL] // Proc. of EMNLP. 2015. http://aclweb.org/anthology/D15-1141.

[143] Sun X, Zhang Y, Matsuzaki T, et al. A Discriminative Latent Variable Chinese Segmenter with Hybrid Word/Character Information[C] // Proc. of NAACL. 2009.

[144] Ferguson J, Durrett G, Klein D. Disfluency Detection with a Semi-Markov Model and Prosodic Features[C/OL] // Proc. of NAACL. 2015. http://www.aclweb.org/anthology/N15-1029.

[145] Mesnil G, Dauphin Y, Yao K, et al. Using Recurrent Neural Networks for Slot Filling in Spoken Language Understanding[J/OL]. IEEE/ACM Transactions on Audio, Speech, and Language Processing, 2015, 23(3) : 530-539. http://dx.doi.org/10.1109/TASLP.2014.2383614.

[146] Zeman D, Ginter F, Hajič J, et al. CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies[C] // Proc. of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies. 2018.

[147] Zhang Y, Nivre J. Transition-based Dependency Parsing with Rich Non-local Features[C] // Proc. of ACL. 2011.

[148] Chen W, Zhang M, Zhang Y. Distributed Feature Representations for Dependency Parsing[J/OL]. IEEE/ACM Transactions on Audio, Speech, and Language Processing, 2015, 23(3) : 451-460. http://dx.doi.org/10.1109/TASLP.2014.2365359.

[149] Bowman S R, Pavlick E, Grave E, et al. Looking for ELMo's friends: Sentence-Level Pretraining Beyond Language Modeling[J/OL]. CoRR, 2018, abs/1812.10860. http://arxiv.org/abs/1812.10860.

[150] Zhang K, Bowman S. Language Modeling Teaches You More than Translation Does: Lessons Learned Through Auxiliary Syntactic Task Analysis[C/OL] // Proc. of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP. 2018. http://www.aclweb.org/anthology/W18-5448.

[151] Tenney I, Xia P, Chen B, et al. What do you learn from context? Probing for sentence structure in contextualized word representations[C/OL] // International Conference on Learning Representations. 2019. https://openreview.net/forum?id=SJzSgnRcKX.

[152] Peters M, Ruder S, Smith N A. To Tune or Not to Tune? Adapting Pretrained Representations to Diverse Tasks[J/OL]. CoRR, 2019, abs/1903.05987. http://arxiv.org/abs/1903.05987.

[153] Straka M. UDPipe 2.0 Prototype at CoNLL 2018 UD Shared Task[C/OL] // Proc. of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies. 2018. https://www.aclweb.org/anthology/K18-2020.

[154] Kondratyuk D. 75 Languages, 1 Model: Parsing Universal Dependencies Universally[J/OL]. CoRR, 2019, abs/1904.02099. http://arxiv.org/abs/1904.02099.

[155] Bentz C, Ruzsics T, Koplenig A, et al. A Comparison Between Morphological Complexity Measures: Typological Data vs. Language Corpora[C] // Proc. of the Workshop on Computational Linguistics for Linguistic Complexity (CL4LC). 2016.

[156] Dryer M S, Haspelmath M. WALS Online[M]. Leipzig : Max Planck Institute for Evolutionary Anthropology, 2013.

[157] Ma X, Hu Z, Liu J, et al. Stack-Pointer Networks for Dependency Parsing[C/OL] // Proc. of ACL. 2018. https://www.aclweb.org/anthology/P18-1130.

[158] McDonald R, Nivre J. Characterizing the Errors of Data-Driven Dependency Parsing Models[C] // Proc. of EMNLP. 2007.

[159] Liang P. Semi-supervised learning for natural language[D]. [S.l.] : Citeseer, 2005.

[160] Ballesteros M, Dyer C, Smith N A. Improved Transition-based Parsing by Modeling Characters instead of Words with LSTMs[C] // Proc. of EMNLP. 2015.

[161] Hinton G E, Vinyals O, Dean J. Distilling the Knowledge in a Neural Network[J]. CoRR, 2015, abs/1503.02531.

[162] Collins M, Roark B. Incremental Parsing with the Perceptron Algorithm[C/OL] // Proc. of ACL. 2004. http://dx.doi.org/10.3115/1218955.1218970.

[163] Liang P, Bouchard-Côté A, Klein D, et al. An End-to-End Discriminative Approach to Machine Translation[C/OL] // Proc. of ACL. 2006. http://dx.doi.org/10.3115/1220175.1220271.

[164] Zhang Y, Clark S. A Tale of Two Parsers: Investigating and Combining Graph-based and Transition-based Dependency Parsing[C] // Proc. of EMNLP. 2008.

[165] Huang L, Fayong S, Guo Y. Structured Perceptron with Inexact Search[C] // Proc. of NAACL. 2012.

[166] Goodman J, Vlachos A, Naradowsky J. Noise reduction and targeted exploration in imitation learning for Abstract Meaning Representation parsing[C] // Proc. of ACL. 2016.

[167] Goldberg Y, Nivre J. A Dynamic Oracle for Arc-Eager Dependency Parsing[C] // Proc. of COLING. 2012.

[168] Ross S, Bagnell D. Efficient Reductions for Imitation Learning[C] // Proc. of AISTATS : Vol 9. 2010.

[169] Ross S, Gordon G, Bagnell D. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning[C] // Proc. of AISTATS : Vol 15. 2011.

[170] Dietterich T G. An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization[J/OL]. Machine Learning, 2000, 40(2) : 139-157. http://dx.doi.org/10.1023/A:1007607513941.

[171] Bengio S, Vinyals O, Jaitly N, et al. Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks[G] // NIPS 28. 2015 : 1171-1179.

[172] Daumé III H, Langford J, Marcu D. Search-based Structured Prediction[J/OL]. Machine Learning, 2009, 75(3). http://dx.doi.org/10.1007/s10994-009-5106-x.

[173] Doppa J R, Fern A, Tadepalli P. HC-Search: A Learning Framework for Search-based Structured Prediction[J/OL]. J. Artif. Intell. Res. (JAIR), 2014, 50. http://dx.doi.org/10.1613/jair.4212.

[174] Vlachos A, Clark S. A New Corpus and Imitation Learning Framework for Context-Dependent Semantic Parsing[J]. Transactions of the Association for Computational Linguistics, 2014, 2 : 547-559.

[175] Chang K-W, Krishnamurthy A, Agarwal A, et al. Learning to Search Better than Your Teacher[C] // Proc. of ICML. 2015.

[176] Buciluǎ C, Caruana R, Niculescu-Mizil A. Model Compression[C/OL] // Proc. of KDD. 2006. http://dx.doi.org/10.1145/1150402.1150464.

[177] Ba J, Caruana R. Do Deep Nets Really Need to be Deep?[G] // NIPS 27. 2014 : 2654-2662.

[178] Nilsson J, Nivre J. MaltEval: an Evaluation and Visualization Tool for Dependency Parsing[C] // Proc. of LREC. 2008.

[179] Ballesteros M, Goldberg Y, Dyer C, et al. Training with Exploration Improves a Greedy Stack LSTM Parser[C] // Proc. of EMNLP. 2016.

[180] Andor D, Alberti C, Weiss D, et al. Globally Normalized Transition-Based Neural Networks[C] // Proc. of ACL. 2016.

[181] Buckman J, Ballesteros M, Dyer C. Transition-Based Dependency Parsing with Heuristic Backtracking[C] // Proc. of EMNLP. 2016.

[182] Kuncoro A, Ballesteros M, Kong L, et al. Distilling an Ensemble of Greedy Dependency Parsers into One MST Parser[C] // Proc. of EMNLP. 2016.

[183] Kuncoro A, Ballesteros M, Kong L, et al. What Do Recurrent Neural Network Grammars Learn About Syntax?[C] // Proc. of EACL. 2017.

[184] Goldberg Y, Sartorio F, Satta G. A Tabular Method for Dynamic Oracles in Transition-Based Parsing[J]. TACL, 2014, 2.

[185] Keskar N S, Mudigere D, Nocedal J, et al. On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima[J]. CoRR, 2016, abs/1609.04836.

[186] Freitag M, Al-Onaizan Y, Sankaran B. Ensemble Distillation for Neural Machine Translation[J]. CoRR, 2017, abs/1702.01802.

[187] Osband I, Blundell C, Pritzel A, et al. Deep Exploration via Bootstrapped DQN[G] // NIPS 29. 2016 : 4026-4034.

[188] Li Z, Zhang M, Chen W. Ambiguity-aware Ensemble Training for Semi-supervised Dependency Parsing[C] // Proc. of ACL. 2014.

[189] Stahlberg F, Byrne B. Unfolding and Shrinking Neural Machine Translation Ensembles[C] // Proc. of EMNLP. 2017.

攻读博士学位期间发表的论文及其他成果

（一）发表的学术论文

[1] **Yijia Liu**, Wanxiang Che, Yuxuan Wang, Bo Zheng, Bing Qin, and Ting Liu. Deep Contextualized Word Embeddings for Universal Dependency Parsing[J] // Transactions on Asian and Low-Resource Language Information Processing.（TALLIP，CCF C 类期刊，已录用待发表）

[2] **Yijia Liu**, Wanxiang Che, Huaipeng Zhao, Bing Qin, and Ting Liu. Distilling Knowledge for Search-based Structured Prediction[C] // Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics. 2018.（ACL，CCF A 类会议）

[3] **Yijia Liu**, Wanxiang Che, Jiang Guo, Bing Qin, and Ting Liu. Exploring Segment Representations for Neural Segmentation Models[C] // Proceedings of 25th International Joint Conference on Artificial Intelligence. 2016.（IJCAI，CCF A 类会议）

[4] **Yijia Liu**, Wanxiang Che, Bing Qin, and Ting Liu. HC-search for Incremental Parsing[C] // Proceedings of 25th International Joint Conference on Artificial Intelligence. 2016.（IJCAI，CCF A 类会议）

[5] **Yijia Liu**, Wanxiang Che, Bo Zheng, Bing Qin, and Ting Liu. An AMR Aligner Tuned by Transition-based Parser[C] // Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. 2018.（EMNLP，CCF B 类会议）

[6] **Yijia Liu**, Yue Zhang, Wanxiang Che, and Ting Liu. Domain Adaptation for CRF-based Chinese Word Segmentation using Free Annotations[C] // Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing. 2014.（EMNLP，CCF B 类会议）

[7] **Yijia Liu**, Yi Zhu, Wanxiang Che, Bing Qin, Nathan Schneider, and Noah A. Smith. Parsing Tweets into Universal Dependency[C] // Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 2018.（NAACL，CCF C 类会议）

[8] **Yijia Liu**, Yue Zhang, Wanxiang Che, and Ting Liu. Transition-Based Syntactic Linearization[C] // Proceedings of the 2015 Conference of the North American Chapter

of the Association for Computational Linguistics: Human Language Technologies. 2015.（NAACL，CCF C 类会议）

[9] Yutai Hou, **Yijia Liu**, Wanxiang Che and Ting Liu, Sequence-to-Sequence Data Augmentation for Dialogue Language Understanding[C] // Proceedings of the 27th International Conference on Computational Linguistics. 2018.（COLING，CCF B 类会议）

[10] Haoyang Wen, **Yijia Liu**, Wanxiang Che, Libo Qin and Ting Liu, Sequence-to-Sequence Learning for Task-oriented Dialogue with Dialogue State Representation[C] // Proceedings of the 27th International Conference on Computational Linguistics. 2018.（COLING，CCF B 类会议）

（二）申请及已获得的专利（无专利时此项不必列出）

[1] 车万翔、**刘一佳**、赵妍妍、刘挺. 一种中文分词增量学习方法：中国，CN201510604035.0[P]. 2015-11-18.

（三）参与的科研项目及获奖情况

[1] 刘挺、车万翔、胡郁、秦兵、陈毅恒、胡国平、张宇、赵妍妍、马汉君、张伟男、**刘一佳**. 语言技术平台及其应用. 黑龙江省科学技术一等奖. 2016.

[2] Wanxiang Che, **Yijia Liu**, Yuxuan Wang, Bo Zheng, and Ting Liu. Towards Better UD Parsing: Deep Contextualized Word Embeddings, Ensemble, and Treebank Concatenation[C] // Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies （CoNLL 2018 通用多国语依存句法分析评测，LAS 第一）

哈尔滨工业大学学位论文原创性声明和使用权限

学位论文原创性声明

本人郑重声明：此处所提交的学位论文《基于动态上下文相关词向量的句子级语言分析技术研究》，是本人在导师指导下，在哈尔滨工业大学攻读学位期间独立进行研究工作所取得的成果，且学位论文中除已标注引用文献的部分外不包含他人完成或已发表的研究成果。对本学位论文的研究工作做出重要贡献的个人和集体，均已在文中以明确方式注明。

作者签名：　　日期：　　年　　月　　日

学位论文使用权限

学位论文是研究生在哈尔滨工业大学攻读学位期间完成的成果，知识产权归属哈尔滨工业大学。学位论文的使用权限如下：

（1）学校可以采用影印、缩印或其他复制手段保存研究生上交的学位论文，并向国家图书馆报送学位论文；（2）学校可以将学位论文部分或全部内容编入有关数据库进行检索和提供相应阅览服务；（3）研究生毕业后发表与此学位论文研究成果相关的学术论文和其他成果时，应征得导师同意，且第一署名单位为哈尔滨工业大学。

保密论文在保密期内遵守有关保密规定，解密后适用于此使用权限规定。

本人知悉学位论文的使用权限，并将遵守有关规定。

作者签名：　　日期：　　年　　月　　日

导师签名：　　日期：　　年　　月　　日

致 谢

本文是在导师秦兵教授与副导师车万翔教授的共同指导下完成的。没有两位老师对于文章内容的编排以及对于文章进度的督促，本文是无法在保证质量的情况下按时完成的。同时，刘挺教授及其领导的哈工大社会计算与信息检索研究中心也为本文的完成提供了充分的支持。具体到文章内容，本文第3章内容得到了麻省理工学院的郭江研究员的指导与帮助。本文第4章的实验是在王宇轩与郑博两位师弟的共同努力下完成的。本文第5章的实验是在赵怀鹏师弟的帮助下完成的。除此之外，天津大学的张梅山副教授对于本文第4章，文灏洋师弟对于本文第2与4章分别给出了宝贵的意见。本文的完成与各位同仁的辛勤劳动是密不可分的。

除了博士论文关注的内容，本文作者博士期间的相关研究也得到了多位同仁的帮助指导。其中包括苏州大学的李正华副教授、西湖大学的张岳副教授、美国华盛顿大学的 Noah A. Smith 副教授、美国乔治华盛顿大学的 Nathan Schneider 助理教授、剑桥大学的朱懿以及王少磊、侯宇泰、覃立波、文灏洋四位师弟。

自本文作者于 2011 年进入实验室开始科研工作，导师秦兵教授给予本文作者多方面的关怀。其中既有对科研的指导，也有对生活的鼓励。这些指导与鼓励使本文作者感受到家的温暖，更加地热爱这个集体。

自 2008 年考入哈工大计算机学院到 2019 年提交博士论文，本文作者的成长过程也得到了副导师车万翔教授的巨大帮助。2008 年秋，车教授任教高级语言程序设计并给本文作者当年的唯一满分。此事培养了本文作者对于计算机的兴趣与自信。2011 年，车教授介绍本文作者进入社会计算与信息检索研究中心开展科研工作，并将实验室的重点项目 — 语言技术平台的开发工作交付给本文作者，使其对于语言分析技术有了深入的了解与认识。2013 年与 2016 年，车教授又分别推荐本文作者去新加坡、美国交流访学，开阔了其研究思路与视野。

实验室主任刘挺教授也从多方面对本文作者进行了指导。刘挺教授的言传身教塑造了本文作者的基本学术价值观。同时，在本文作者开始科研工作之初，刘挺教授强调的 ‘‘一万小时’’、‘‘成为专家” 的观点都对本文作者产生了积极的影响。刘挺教授也积极为包括本文作者在内的实验室成员创造学习交流机会。在与毕业师兄交流中，本文作者学得的时间管理技术并受益良多。

2013 年到 2014 年，本文作者访问新加坡科学与设计大学，在张岳副教授的指导下进行了一年的科研工作。这段访问经历从基本科研方法、论文写作等方面锻炼了本文作者，对于本文作者的快速成长有重大意义。

2016 年到 2017 年，本文作者访问美国华盛顿大学并得到 Noah A. Smith 教授

的指导。Smith 教授帮助本文作者更好地理解了自然语言处理与机器学习之间的关系。更重要的是，通过与 Smith 教授的交流，本文作者认识到学术合作的重要性。

本文作者也要感谢实验室的张宇教授，刘铭副教授，丁效老师以及冯骁骋老师。几位老师在工作生活中对本文作者给出有效的建议。同时，本文作者也要感谢包括牛国成、韩冰、徐伟、徐梓翔、刘洋等等共同奋斗的同学。

最后，本文作者需要感恩父母给予其生命，养育其成人，支持其完成博士学业。

个人简历

刘一佳，男，1988 年 8 月，出生于辽宁省本溪市。

教育经历

• 2012 年 9 月至今：哈尔滨工业大学计算机学院社会计算与信息检索研究中心攻读工学博士学位，导师为秦兵教授，副导师为车万翔教授；

• 2012 年 9 月 — 2014 年 7 月：哈尔滨工业大学计算机学院社会计算与信息检索研究中心获得工学硕士学位，导师为车万翔副教授；

• 2008 年 9 月 — 2012 年 7 月：哈尔滨工业大学计算机学院获得工学学士学位。

实习经历

• 2016 年 9 月 — 2017 年 9 月：美国华盛顿大学（University of Washington），导师为 Noah A. Smith 副教授；

• 2013 年 10 月 — 2014 年 10 月：新加坡科学与设计大学（Singapore University of Technology and Design）访问，导师为张岳助理教授；

• 2011 年 7 月 — 2011 年 11 月：北京百度公司自然语言处理部门实习。

获奖情况

• 2018 年 12 月：百度奖学金；

• 2016 年 9 月：华为奖学金（研究生阶段）；

• 2013 年 9 月：国家奖学金（研究生阶段）；

• 2010 年 10 月：2010 年国际大学生程序设计竞赛亚洲区赛杭州站，银奖；

• 2010 年 9 月：华为奖学金（本科阶段）。

项目经历

• 2018 年 4 月 — 2018 年 6 月：CoNLL 2018: 多语言通用依存分析评测；

• 2013 年至今：语言技术平台。