第3章

基于多特征融合和语义增强的软件知识实体抽取方法

3.1 引言

近年来，StackOverflow、Lampcms[①]、Learnku[②]、思否[③]、掘金[④]、德问[⑤]等软件知识社区得到了快速发展，成为广大软件开发人员交流、共享软件知识的平台和重要的软件领域知识库[1]。通过这些软件知识社区，软件开发人员可以根据个人需求进行提问、回答和评论等操作，帮助软件开发人员快速掌握特定软件知识，解决开发过程中遇到的问题。以软件知识社区 StackOverflow 为例，近期该社区拥有600 多万注册用户，发布了约 2100 万个软件工程领域问题，在这些问题的回答、评论、链接等交互信息中蕴含着海量的软件知识实体。

但是，传统基于关键字、主题模型等文本处理技术把软件知识实体当作普通文本处理，没有关注到软件知识社区文本的社会化特征和专业领域特征，无法适应软件开发人员获取软件知识的需求。如何从软件知识社区中准确、高效地获取高质量的软件知识，成为推进智能化软件工程的重要因素，也成为大数据知识工程在软

① http://www.lampcms.com/.
② https://learnku.com/.
③ https://segmentfault.com/.
④ https://juejin.cn/.
⑤ http://www.dewen.net.cn/.

件领域的重要挑战。

知识图谱作为一种知识表示形式，用图结构来建模实体、概念及其相互之间的语义关系，能增强对知识组织结构的表达，使用户能快速、准确、智能地处理各类信息[9]。对软件知识社区的用户生成文本所蕴含的软件知识实体及其关系信息进行识别和抽取，进而构建软件知识图谱将会促进智能问答、软件文档生成和专家推荐等以实体为中心的智能化软件工程的发展。

实体抽取旨在文本中识别并分类所属预定义的语义类型（如人名、地名和组织名）的名词性指称项，是知识图谱构建和自然语言理解的重要前提基础[8]。我们所指的软件知识实体抽取是指从海量非结构化的软件知识社区文本中识别和抽取有关软件编程、软件开发库和软件项目等软件工程领域实体，并按其语义分类到预先定义的实体类型中。由于软件知识社区文本是用户生成的非结构化短文本，具有非常强的社会化和专业领域特征，普遍存在如下问题：

（1）遵循的编程语言规范不统一或拼写不规范，造成大量拼写错误和简写现象，导致实体变体现象。例如，软件知识实体“JavaScript”存在名称缩写和拼写错误产生的实体“JS”“javascripte”等。

（2）某些特定软件实体名称为常用词，造成实体稀疏问题。

（3）同一实体词在不同语境上下文中可以归属不同实体类型，造成实体歧义问题。比如，软件知识实体“Mac”可以标记为“PlatCOS”（操作系统），也可以标记为“SLMDL”（移动开发库）。

（4）存在少见、独特的软件知识实体，造成 Out-of-Vocabulary 单词无法识别问题，如“Glass-box unit-testing technique”“application framework”等。

另外，软件知识实体抽取任务缺乏统一、完备的实体类型定义及公开的领域标注数据集。

因此，上述存在的问题给软件知识实体抽取任务带来了实体歧义、实体变体、无法识别未登录词和缺乏领域标注数据集等挑战。同时，作为特定领域知识图谱构建的关键基础，软件知识实体抽取结果的可信度要求更高，需要改进算法提升软件知识实体抽取的准确率。

针对上述软件知识实体抽取存在的诸多挑战，我们综合考虑单词、句法、实体上下文及其语义特征，提出一种基于多特征融合和语义增强的软件知识实体抽取方法。一方面，从句子序列建模的角度，使用 BiLSTM 模型和 GCN 模型，将上下文特征和句法依存特征进行融合，从而得到多特征融合的句子序列上下文语义表示；另一方面，从实体建模的角度，利用一个基于注意力权重的实体语义增强策略，获取领域实体的增强特征表示。该方法区别于当前通用领域的实体抽取方法，它对模型的输入嵌入层、特征融合层和特征增强层进行了改进。具体地，主要工作

如下：

（1）针对实体歧义问题，提出一种基于 BiLSTM 模型和 GCN 模型的混合神经网络模型，实现句子序列的多特征融合表示。具体而言，在模型的输入嵌入层，利用 BERT 模型获取高质量的软件工程领域词向量表示；在特征融合层，利用 BiLSTM 模型和 GCN 模型构建特征编码器，捕获句子序列的上下文特征和句法依存特征。

（2）针对实体变体和无法识别未登录词问题，在模型的特征增强层提出了一个基于注意力权重的实体语义增强策略，以捕获领域实体的语义增强特征表示。

（3）针对软件工程领域缺乏标注数据集的问题，基于软件知识社区 StackOverflow 的问答文本构建了涵盖 12751 个句子、515943 个 tokens 和 40 个实体类型的软件工程领域标注数据集。

3.2 相关工作

当前，研究人员利用机器学习和深度学习方法开展软件工程领域实体抽取方法研究。Ye 等[13]提出了一种基于半监督学习方法的软件工程领域命名实体识别方法 S-NER，将软件知识实体类型分为 Programming Languages、Platform、API、Tool-library-framework、Software Standard 五个类型，并利用人工特征工程将拼写特征、词法上下文特征、词位特征和词典特征进行融合，获得了较好的性能。它属于较早关注从知识社区文本中抽取软件知识的研究工作，为后续研究工作打开了思路，但存在对繁杂的人工特征工程和高质量标注数据集过度依赖的问题。Reddy 等[105]在上述工作的基础上，利用 BiLSTM 和 CRF 进行软件工程领域实体抽取，并对预定义实体类型进行扩展，取得了较好的性能。针对非结构化数据、半结构化数据和代码数据，Lv 等[106]分别利用 BiLSTM+CRF、模板匹配和抽象句法树从软件知识社区抽取软件知识实体，并采用基于 TF-IDF 的关键字抽取、TextRank、K-Means 等方法解决标注数据集缺乏的问题。Tabassum 等[107]结合 StackOverflow 的问答文本数据，构建了包含 15372 个句子和 20 个实体细分类的计算机编程领域语料库，并提出一个基于注意力机制的实体识别模型，提高了代码实体识别效果。

Sun 等[108]提出了一种基于 BERT 词嵌入的软件实体识别方法。该方法首先利用 BiLSTM-CRF 模型和词向量嵌入技术构建实体识别模型，然后通过引入 BERT 预训练语言模型对模型输入层的词向量进行优化，最后以软件知识社区 StackOverflow 为案例进行了实验验证。为了应对软件实体识别受限于小实体词

汇表或噪声训练数据的影响，Tai 等[109]利用维基百科分类法开发了一个全面实体词典，包括 12 种细粒度类型的 79000 个软件实体，以及超过 170 万个句子的大型标记数据集；同时，提出了一种基于自正则化的学习方法来训练软件实体识别模型，在 Wikipedia 基准测试和两个 StackOverflow 基准测试上取得了较好的性能。

针对 Bug 文本包含代码、缩写和特定软件词汇的情况，现有命名实体识别方法无法直接应用于 Bug 实体识别的问题，Zhou 等[110]提出了一种基于条件随机场模型和词嵌入技术的 Bug 实体识别方法 BNER，并在两个开源项目（Mozilla 和 Eclipse）上构建了一个基线语料库。为进一步改善 Bug 实体识别的效果，Zhou 等[111]提出一种基于 BiLSTM 和 CRF 的 Bug 实体识别模型 DBNER。该模型从海量 Bug 报告数据中提取多种特征，并利用注意力机制提高了 Bug 报告中实体标签的一致性。

针对从软件开发需求文本中提取需求实体存在的问题，Li 等[112]提出了一种利用 LSTM-CRF 模型进行需求实体提取，并引入通用知识减少对标签数据的依赖的新方法 RENE。在模型构建阶段，RENE 构建了一个 LSTM-CRF 模型和一个用于迁移学习的同构 LSTM 语言模型；在 LSTM 语言模型训练阶段，RENE 捕获通用知识并适应需求上下文；在 LSTM-CRF 训练阶段，RENE 用迁移层对模型进行训练；在需求实体抽取阶段，RENE 将训练好的模型应用于新出现的需求，并自动抽取需求实体。在隐私需求工程领域，Herwanto 等[113]利用实体识别模型以及上下文嵌入技术开发了一种检测用户故事中与隐私相关需求实体的自动化方法。该方法将实体分为数据主体实体、处理实体和个人数据实体，在精确率和召回率方面取得了较好的性能。

以上方法利用深度学习模型自动提取文本特征，减少了人工耗时操作，但由于多以领域关键字作为实体进行抽取，缺乏对特定软件工程领域的实体歧义、实体变体、无法识别未登录词等问题的关注，软件知识实体抽取的质量有待提升。

3.3　模型与方法

3.3.1　任务建模与方法分析

我们所指的软件知识实体抽取任务是从非结构化的软件知识社区文本中自动识别软件知识实体，并根据预先定义的实体类型进行分类。因此，软件知识实体抽取任务可以形式化定义为一个 4 元组 $SE=(X,Y_e,\delta,NA)$，其中：

X 为软件知识社区文本的句子序列，$X=(x_1,x_2,\cdots,x_n)$；

$Y_e(e_i)$为预测候选实体 e_i 类型的函数，$Y_e(e_i)\in\delta\cup\{NA\}$，并产生候选软件

知识实体集 $E=(e_1,e_2,\cdots,e_{|E|})$，$e_i=(x_i,x_{i+1},\cdots,x_{i+k})$；

δ 为预定义的实体类型集合；

NA 表示非实体。

例如，给定软件知识社区文本的句子序列“*GetHashCode is method of base Object class of .Net Framework.*”，软件知识实体抽取任务的目标是准确识别出实体“*GetHashCode*”和实体“*.Net Framework*”，并分类到正确的实体类型。

结合以上任务目标，我们提出一个基于社区文本的软件知识实体抽取模型，命名为 AS-SNER，其由输入嵌入层、特征融合层、特征增强层和标签解码层组成，整体架构如图 3-1 所示。

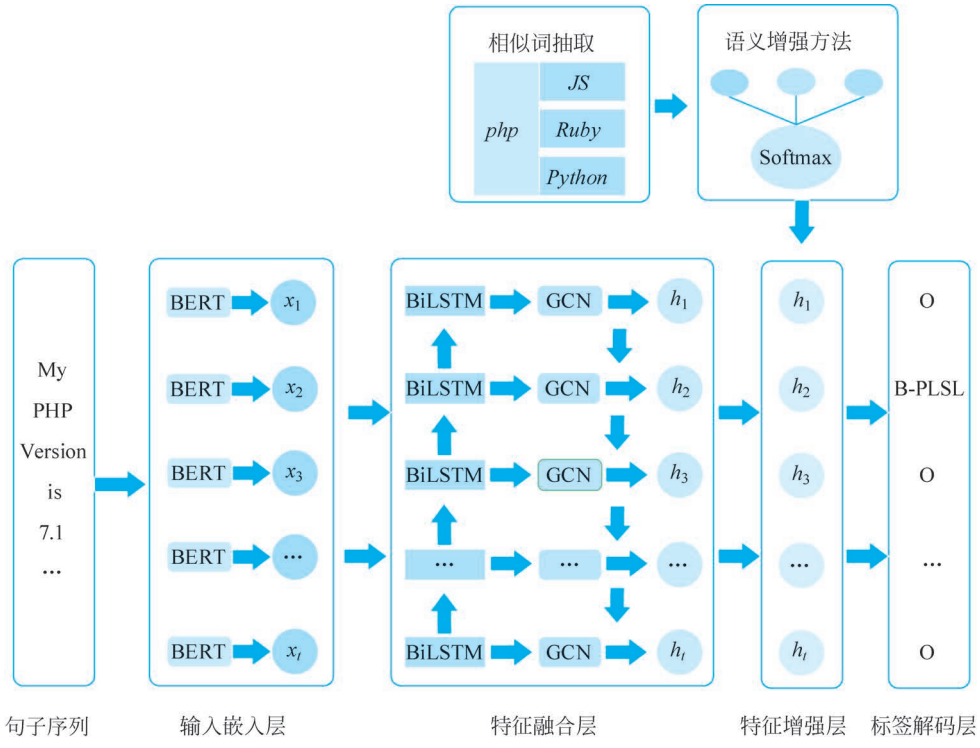

图 3-1 AS-SNER 模型整体架构图

AS-SNER 模型中各层的主要功能如下：

在输入嵌入层，为了获取高质量的输入句子序列的词向量表示，利用 BERT 模型对软件知识社区文本进行无监督学习，并将其作为特征编码器捕获句子序列的文本特征，进而丰富软件知识实体抽取模型的输入特征表示。

在特征融合层，由于句子序列的上下文信息和句法依存信息对实体抽取任务

具有关键作用，使用 BiLSTM 模型和 GCN 模型构建特征编码器，分别对句子序列的上下文信息和句法依存信息进行特征编码，获取句子序列的多特征表示，从而缓解实体歧义问题。

在特征增强层，针对实体变体和无法识别未登录词的问题，提出一个基于注意力权重的实体语义增强策略，并通过特征融合获取特定领域实体的增强特征向量表示。

在标签解码层，使用 CRF 模型构建标签解码器，获取全局最优的标签序列。

3.3.2　输入嵌入层

模型的输入嵌入层负责将软件知识社区文本中的句子序列转换为低维、稠密的分布式向量表示，并输入模型的下一层。在输入嵌入层，获取丰富的领域文本特征对提升实体抽取的质量具有重要作用[114,115]。BERT 模型作为一种预训练语言模型，可以作为特征编码器捕获文本的语义信息，生成符合当前语境的动态词向量，进而提升模型的语义消歧能力。BERT 模型编码句子序列的过程描述：首先对于软件知识社区文本的输入句子序列 $X=(x_1,x_2,\cdots,x_n)$，经过基于块的分词、句子序列的首尾添加标识符[CLS]和[SEP]后，得到句子序列对应的 Token 序列；然后针对 Token 序列的每个 Token 产生 Token 向量、分割向量和位置向量，三个向量经过求和后，得到 BERT 模型的输入向量表示；经过特征编码后，得到句子序列 X 的动态词向量表示，即

$$\boldsymbol{H}=[h_1,h_2,\cdots,h_n]=\mathrm{BERT}(x_1,x_2,\cdots,x_n) \tag{3.1}$$

为了获取句子序列的高质量词向量表示，提升从软件知识社区文本中抽取实体的性能，我们利用 BERT 模型对海量软件知识社区的文本进行预训练，并将获得的精调预训练语言模型 SWBERT 作为模型的输入特征编码器，详细内容见 3.4.1 节。

3.3.3　特征融合层

软件知识社区文本存在不同语境导致的实体歧义问题和多个单词构成实体名称导致的长距离依赖问题，因此，对于基于社区文本的软件知识实体抽取任务，捕获句子序列的上下文特征和句法依存特征，并融合成句子序列的领域多特征表示，能提升模型的消歧能力和长距离信息处理能力。

1. 基于 BiLSTM 模型的上下文特征编码

在软件知识实体抽取过程中，数据是来自软件知识社区文本的句子序列，适合使用时序模型对其进行建模，以捕获句子序列的全局上下文信息。由 Hochreiter[116] 提出的 LSTM 网络利用门控机制设计单元结构，可以选择性地保留或者丢弃句子序列的有用或无用信息，提升建模句子序列上下文语义的能力。LSTM 网络的循环

单元结构如图 3-2 所示。

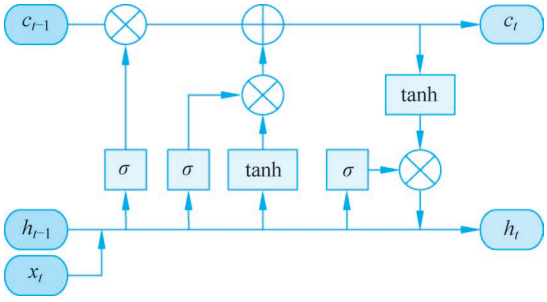

图 3-2 LSTM 网络的循环单元结构

模型 t 时刻的形式化表示如下[116]：

$$i_t=\sigma(\boldsymbol{W}_{xi}x_t+\boldsymbol{W}_{hi}h_{t-1}+\boldsymbol{W}_{ci}c_{t-1}+b_i) \tag{3.2}$$

$$f_t=\sigma(\boldsymbol{W}_{xf}x_t+\boldsymbol{W}_{hf}h_{t-1}+\boldsymbol{W}_{cf}c_{t-1}+b_f) \tag{3.3}$$

$$c_t=f_tc_{t-1}+i_t\tanh(\boldsymbol{W}_{xc}x_t+\boldsymbol{W}_{hc}h_{t-1}+b_c) \tag{3.4}$$

$$o_t=\sigma(\boldsymbol{W}_{xo}x_t+\boldsymbol{W}_{ho}h_{t-1}+\boldsymbol{W}_{co}c_t+b_o) \tag{3.5}$$

$$h_t=o_t\tanh(c_t) \tag{3.6}$$

式中：σ 和 tanh 表示非线性激活函数；i_t 表示输入门；f_t 表示遗忘门；c_t 表示细胞单元；o_t 表示输出门；x_t 表示输入单元；h_t 表示隐层状态；$\boldsymbol{W}$ 表示权重矩阵，b 表示偏置项。

由 LSTM 模型的结构可知，在句子序列的处理过程中，由 x_t、f_t、c_t 和 o_t 的状态捕获当前句子序列的历史信息（即上文信息），但缺乏未来信息（即下文信息），而下文信息对软件知识实体抽取任务同样具有重要作用。所以，利用两个方向相反的 LSTM 模型构建一个双向 LSTM 模型，可以同时捕获当前 t 时刻句子序列的上下文信息，最终输出由正、反两个方向的 LSTM 模型输出拼接获得，模型结构如图 3-3 所示。

双向 LSTM 模型的计算过程如下：

在模型的 t 时刻，正向 LSTM 的隐层状态的输出表示为

$$\overrightarrow{\boldsymbol{h}}_t=\mathrm{LSTM}(x_t,\overrightarrow{\boldsymbol{h}}_{t-1}) \tag{3.7}$$

式中：$\overrightarrow{\boldsymbol{h}}_t$ 表示 t 时刻的输出；x_t 表示 t 时刻的输入；$\overrightarrow{\boldsymbol{h}}_{t-1}$ 表示 $t-1$ 时刻的隐层状态。反向 LSTM 的隐层状态的输出表示为

$$\overleftarrow{\boldsymbol{h}}_t=\mathrm{LSTM}(x_t,\overleftarrow{\boldsymbol{h}}_{t+1}) \tag{3.8}$$

式中：$\overleftarrow{\boldsymbol{h}}_t$ 表示 t 时刻的输出；x_t 表示 t 时刻的输入；$\overleftarrow{\boldsymbol{h}}_{t+1}$ 表示 $t+1$ 时刻的隐层状态。

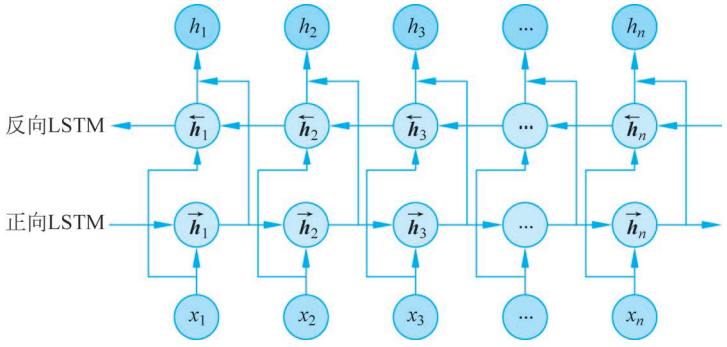

图 3-3　双向 LSTM 模型结构

因此，双向 LSTM 模型在 t 时刻的总输出为

$$\boldsymbol{h}_t = [\overrightarrow{\boldsymbol{h}}_t ; \overleftarrow{\boldsymbol{h}}_t] \tag{3.9}$$

2. 基于 GCN 模型的句法依存特征编码

句法特征是指通过句法分析技术，提取句子的组成结构和单词间依存关系等句法层面的特征。其中，句法依存关系特征着重刻画句子序列局部及其相互之间的依赖关系。因此，获取句子序列单词间的依存关系特征，能捕获领域实体的长距离依赖特征，有助于识别和发现专业领域的潜在实体。由于软件知识社区文本是用户生成文本，存在大量的特定编程语言规范的短语实体和多个单词构成的合成词实体，获取单词间的句法依存特征对软件知识实体抽取具有重要作用。

图卷积神经网络是一种基于图结构数据的卷积神经网络，通过对图节点及边进行建模，捕捉节点间的依赖关系，逐渐被应用于自然语言处理领域，如文本分类[117]、语义角色标注[118]、关系抽取[119]和机器翻译[120]。由于标准的 LSTM 网络无法建模句子序列的句法依存结构信息，使用 GCN 模型构建特征编码器，对句子序列的句法依存特征进行编码，将句子序列中每个单词视为节点，节点通过汇集其相邻节点的信息生成该节点的特征向量表示。具体过程描述如下：

(1) 句法依存关系分析。

为了捕获句子序列的句法依存特征，需要对其进行句法依存关系分析，识别和挖掘句子序列中单词之间的依存关系，并以图形化的方式进行可视化。

利用斯坦福大学开发的 StanfordCoreNLP 工具[①]对软件知识社区文本的句子序列进行句法依存关系分析，并以句法分析图进行示例说明。

例如，对于软件知识社区的句子序列“*How to convert a TIFF to a JPG with ImageMagick ?* ”，经过句法依存分析处理后，可以得到对应的依存关系图，如图 3-4

① https://stanfordnlp.github.io/CoreNLP/.

所示，其中，节点为句子序列的单词，边为节点间的句法依存关系。

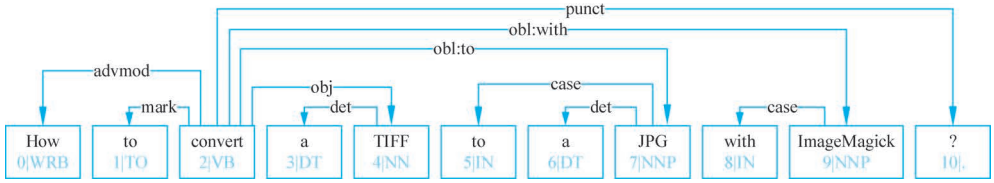

图 3-4 句法依存关系图示例

由图 3-4 可知，该句子序列的实体分别为“*TIFF*”“*JPG*”“*ImageMagick*”，依存词为“*convert*”，其中，“*TIFF*”与依存词“*convert*”的依存关系为“*obj*”，表示实体“*TIFF*”是谓语“*convert*”的宾语。

（2）邻接矩阵构建。

在获取句子序列的依存关系图后，可以利用 GCN 对句法依存关系的结构信息进行建模。给定图卷积神经网络 $G=(V,E,\boldsymbol{A})$，其中：V 表示节点，即句子序列中的所有单词；E 表示不考虑方向情况下的边，即节点之间的关系；$\boldsymbol{A}$ 表示对应的邻接矩阵。邻接矩阵反映了句子序列中单词之间的句法依存关系，其构建过程形式化描述如下：

给定句子序列 $X=(x_1,x_2,\cdots,x_n)$，x_i 为单词节点，若 x_i 到 x_j 之间存在依存关系，表示 x_i 到 x_j 之间存在边（无向边），则 $A_{ij}=A_{ji}=1$；否则，$A_{ij}=A_{ji}=0$。同时，为了聚合节点自身的信息，每个节点添加了自循环，即 $A_{ii}=1$。由此，将含有 t 个单词的句子序列表示为一个 $t\times t$ 的矩阵 $\tilde{\boldsymbol{A}}$。

例如，上述句子序列“*How to convert a TIFF to a JPG with ImageMagick ?*”的邻接矩阵 $\tilde{\boldsymbol{A}}$ 为

$$\tilde{\boldsymbol{A}}=\begin{bmatrix}1&0&0&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0&0&0\\1&1&1&0&1&0&0&1&0&1\\0&0&0&1&0&0&0&0&0&0\\0&0&0&1&1&0&0&0&0&0\\0&0&0&0&0&1&0&0&0&0\\0&0&0&0&0&0&1&0&0&0\\0&0&0&0&0&1&1&1&0&0\\0&0&0&0&0&0&0&0&1&0\\0&0&0&0&0&0&0&0&1&1\end{bmatrix}$$

（3）依存特征提取。

在一个 L 层的 GCN 句法特征编码层中，节点 i 通过图卷积操作聚合该节点相

邻节点的特征，完成特征向量的输出：

$$\boldsymbol{h}_i^l = \sigma(\tilde{\boldsymbol{D}}^{-\frac{1}{2}}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{D}}^{-\frac{1}{2}}\boldsymbol{h}_i^{l-1}\boldsymbol{W}) \tag{3.10}$$

式中：σ 为非线性激活函数；$\tilde{\boldsymbol{A}}$ 为添加自循环的邻接矩阵，$\tilde{\boldsymbol{A}}=\boldsymbol{A}+\boldsymbol{I}$；$\tilde{\boldsymbol{D}}$ 为 $\tilde{\boldsymbol{A}}$ 对应的度矩阵；$\boldsymbol{h}_i^{l-1}$ 为第 l 层节点的输入；$\boldsymbol{h}_i^l$ 为第 l 层节点的输出；$\boldsymbol{W}$ 为权重矩阵。

3.3.4　特征增强层

软件知识社区文本存在大量特定的软件知识实体名称拼写错误、缩写和简写的情况，导致软件知识实体抽取任务面临实体变体和无法识别未登录词的问题。特别是在实体类型复杂的场景下，传统基于领域词典或外部特征的方法，会给软件知识实体抽取结果带来噪声和错误传播的问题，造成软件知识实体标签错误。

相关文献[121]研究表明，增强特定领域实体的语义特征表示，能有效缓解实体变体和无法识别未登录词的问题，提升软件知识实体抽取的准确性。因此，针对软件知识社区的用户生成文本，在模型的特征增强层提出一种基于注意力权重的语义增强策略，增强领域实体的语义特征表示，具体过程如算法 3-1 所示。

算法 3-1　领域实体语义增强算法

Input: 软件知识社区文本的句子序列 $X=(x_1,x_2,\cdots,x_n)$
Output: 句子序列 X 语义增强表示 $H=(h_1,h_2,\cdots,h_n)$

```
Begin
  获取经 GCN 模型编码后的句子序列 X 的向量表示          //式(3.10)
  for x_i from X do
    从 SWBERT 和 SEthesaurus 中抽取 k 个 x_i 相似词 S=(s_1,s_2,…,s_k)
    获取相似词集的向量表示 C=(c_1,c_2,…,c_k)
    for s_i from S do
      计算 s_i 对于 x_i 的语义贡献权重                  //式(3.11)、式(3.12)
      得到 x_i 的语义增强向量表示 A_i                   //式(3.13)
    endfor
    通过向量融合得到 x_i 的语义增强特征表示 h_i         //式(3.14)
  endfor
  返回句子序列 X 的语义增强表示 H=(h_1,h_2,…,h_n)
End
```

由算法 3-1 可知，基于注意力权重的语义增强策略主要包括基于语义一致性的相似词抽取、基于注意力的语义贡献度权重分配和特征向量融合三个步骤。首先引入软件工程领域预训练词向量和软件工程领域词库作为辅助资源，通过语义相似度计算，抽取语义一致性高的领域单词列表；然后利用注意力机制计算相似领域单词对目标词的语义贡献度权重，从而保证语义一致性；最后与隐层向量进行拼接，获得单词的语义增强特征表示。

1. 基于语义一致性的相似词抽取

选择高质量的外部辅助资源对相似词抽取具有重要作用，会影响语义增强表示的质量。一方面，利用 BERT 预训练语言模型对海量的软件知识社区文本的句子序列进行预训练，得到一个面向软件工程领域的精调预训练语言模型 SWBERT（详细内容见 3.3.1 节）。该预训练语言模型 SWBERT 蕴藏着丰富的软件知识实体的语义信息，可以作为相似词抽取的辅助资源库。另一方面，一些研究工作[122,123]关注软件工程领域词库建设，通过无监督学习从 StackOverflow、Wikipedia 等语料库自动构建软件工程领域词库，实现了面向软件工程领域的缩写识别、相似词识别等功能。为了提升相似词抽取的质量，保证语义一致性，选择从软件工程领域精调预训练语言模型 SWBERT 和软件工程领域词库 SEthesaurus[122]抽取单词的相似词作为语义增强辅助。

例如，对于输入句子序列“*My php version is 7.1.*”中的单词“*php*”，通过语义相似度计算，抽取到“*JavaScript*”“*JS*”“*Ruby*”“*Groovy*”“*Cython*”“*Sinatra*”等相似词，这些相似词具有软件领域相关性，将作为辅助资源增强单词“*php*”的语义表示。

2. 基于注意力机制的语义贡献度权重分配

由于是从不同语境抽取相似词，这些相似词对目标词的语义贡献度具有一定的差异性，我们结合注意力机制根据相似词的语义贡献度进行权重分配，以考虑相似词的语义一致性问题。

注意力机制的本质是选择性关注某些重要信息，是一种分配信息处理能力的选择机制，其目标是将某个查询 Query 和一组键值对映射（Key-Value）作为输入，输出每个 Value 所对应的权重[124]，计算公式如下：

$$\mathrm{Attention}(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V})=\sum_{i=1}^{l}\mathrm{Sim}(Q_i,K_i)\cdot V_i \tag{3.11}$$

式中：$\boldsymbol{Q}$、$\boldsymbol{K}$ 和 $\boldsymbol{V}$ 分别为查询、键和值的矩阵表示；Sim()表示相似度计算。

因此，借鉴注意力机制的思想，对于软件知识社区文本的句子序列 $X=(x_1,x_2,\cdots,x_n)$中的每个单词 $x_i\in X$，抽取出 k 个相似词 $S=(s_1,s_2,\cdots,s_k)$，并获取对应的词向量表示 $\boldsymbol{C}=(c_1,c_2,\cdots,c_k)$，则每个相似词 s_i 对于 x_i 的语义贡献权重为

$$w_i=\frac{\exp(\boldsymbol{h}_i\cdot\boldsymbol{c}_i)}{\sum_{j=i}^{k}\exp(\boldsymbol{h}_i\cdot\boldsymbol{c}_j)} \tag{3.12}$$

式中：$\boldsymbol{h}_i$ 为句子序列中单词 x_i 对应的上下文隐层向量，相似度函数采用点积操作进行运算。

在获取每个相似词 s_i 对于 x_i 的贡献权重后，采用加权求和计算当前单词 x_i

的语义增强表示，即

$$\boldsymbol{A}_i = \sum_{i=1}^{n} w_i \boldsymbol{c}_i \tag{3.13}$$

3. 特征向量融合

经过上述两个步骤得到软件知识社区文本的句子序列中当前单词 x_i 的语义增强表示向量 $\boldsymbol{A}_i$，然后与经过多特征编码的单词 x_i 的隐层向量 $\boldsymbol{h}_i$ 进行融合，作为标签解码层的输入向量，即

$$\boldsymbol{h}_i = \boldsymbol{h}_i \oplus \boldsymbol{A}_i \tag{3.14}$$

3.3.5　标签解码层

将软件知识实体抽取过程视为序列标注问题，采用 BIO（Beginning-Inside-Outside）标注模式对语料库的实体信息进行数据标注，其中，“B-”表示软件知识实体开始的位置，“I-”表示对应实体内部，“O”表示非实体。BIO 标注模式的结构示例如表 3-1 所示。

表 3-1　BIO 标注模式的结构示例

实体类型	实体标签	开始标签	内部标签
面向对象语言	PLOO	B-PLOO	I-PLOO
面向过程语言	PLPD	B-PLPD	I-PLPD
脚本语言	PLSL	B-PLSL	I-PLSL
Web 开发语言	PLML	B-PLML	I-PLML
结构查询语言	PLSQL	B-PLSQL	I-PLSQL
指令集	PlatCIS	B-PlatCIS	I-PlatCIS
云计算平台	PlatCCP	B-PlatCCP	I-PlatCCP
Web 服务器	PlatCWSW	B-PlatCWSW	IPlatCWSW
非实体	O	O	O

由 BIO 标注模式的结构可知，标签“B-”、标签“I-”和标签“O”之间并不是彼此独立的，存在相互约束的关系。例如，在句子序列的标注过程中，“I-PLOO”标签可以出现在“B-PLOO”标签之后，形成有效的标签序列“B-APIWA I-APIWA”；但不能在“B-PLOO”标签之前，形成无效的标签序列“I-APIWA B-APIWA”。

因此，在软件知识抽取任务的标签解码层，如果直接使用类似全连接层的方法对实体的类型进行分类预测，会忽略标签“B-”、标签“I-”和标签“O”之间的约束关系，造成标签错误或无效标签的情况。

CRF 模型作为一种条件概率模型，它结合句子序列的局部信息和全局观察信息对句子序列进行建模，能捕获句子序列的标签约束关系，避免标签错误或无效标

签问题。因此，使用 CRF 模型作为标签解码层，通过从句子序列的全局层面考虑相邻标签的约束关系，以避免上述问题。

由此，标签解码层[125]的任务目标可以形式化表示为给定句子序列 $X=(x_1,x_2,\cdots,x_t)$ 的隐层状态序列 $H=(h_1,h_2,\cdots,h_t)$，通过 CRF 模型训练，求解最优标签序列 $Y=(y_1,y_2,\cdots,y_t)$。其具体计算过程如下：

(1) 设 C_{ij} 为句子序列第 j 个单词被预测为标签 i 的概率，计算标签序列的总得分：

$$\text{score}(H,Y)=\sum_{i=1}^{t}Z_{y_{i-1},y_i}+\sum_{i=1}^{t}C_{i,y_i} \tag{3.15}$$

式中：$\boldsymbol{Z}$ 为标签之间的转移矩阵，Z_{y_{i-1},y_i} 表示 y_{t-1} 到 y_t 的转移概率。

(2) 通过归一化指数函数 Softmax 对标签序列 Y 的概率进行归一化：

$$C(Y\mid H)=\frac{\mathrm{e}^{\text{score}(H,Y)}}{\sum\limits_{\omega\in Y(h)}\mathrm{e}^{\text{score}(H,\omega)}} \tag{3.16}$$

(3) 使用动态规划 Viterbi 算法计算得分最高的标签序列：

$$y^{*}=\underset{\omega\in Y(h)}{\text{argmax}}(h,\omega) \tag{3.17}$$

式中：$Y(h)$ 为所有可能的标签序列。

由此，上述基于多特征融合和语义增强的软件知识实体抽取的过程可以形式化表示为算法 3-2。

算法 3-2　基于多特征融合和语义增强的软件知识实体抽取算法

Input: 软件知识社区文本的句子序列 $X=(x_1,x_2,\cdots,x_n)$ 及其句法依存信息，预训练语言模型 SWBERT

Output: 软件知识实体的标签序列 $y=(y_1,y_2,\cdots,y_n)$

```
Begin
  for each epoch do
    for each batch do
      通过 SWBERT 获取句子序列 X 的词向量表示                    //式(3.1)
      通过 BiLSTM 获取句子序列 X 上下文特征表示                  //式(3.2)～式(3.9)
      通过 GCN 获取句子序列的句法依存特征                        //式(3.10)
      调用算法 3-1 获取句子序列的语义增强表示 H=(h1,h2,…,hn)
      计算标签序列 y 的得分                                      //式(3.15)
      归一化标签序列 y 的概率                                    //式(3.16)
      获取最优标签序列 y*                                        //式(3.17)
    endfor
    返回标签序列 y=(y1,y2,…,yn)
  endfor
End
```

由算法 3-2 可知，模型根据超参数的设置进行参数初始化，并根据 batch size 的值设置每个 epoch 中单个批次训练数据的大小。在模型训练过程中，软件知识社区文本的句子序列通过预训练语言模型 SWBERT 获取对应的词向量表示，并输入 BiLSTM 模型和 GCN 获取句子序列的上下文特征和句法依存特征表示。对句子序列的每个单词通过基于注意力权重的语义增强策略获取对应的语义增强表示向量，再经过特征向量融合后得到最终的向量表示。最后，经过标签解码层对标签的概率进行预测，得到句子序列对应的标签序列。

3.4　实验与分析

为验证我们所提出的基于社区文本的软件知识实体抽取方法的实际效果，下面以软件知识社区 StackOverflow 为例开展实验与分析。同时，选择当前实体抽取领域的经典模型作为基线模型进行对比实验，根据实验结果对模型性能进行分析与评价。实验的软件环境基于 Python 语言和深度学习框架 PyTorch，硬件环境为 Intel Core i9-13900K 处理器，3.0GHz 时钟频率，GeForce RTX 4090 GPU，24GiB 显存。

3.4.1　数据集构建

1. 基于 BERT 模型的预训练词向量构建

为了获取高质量的软件工程领域词向量表示，我们使用预训练语言模型 BERT 对软件知识社区文本的海量句子序列进行了无监督训练，以获取软件工程领域的精调预训练语言模型。具体包括以下三个步骤：

(1) 获取数据源。软件知识社区 StackOverflow 的官方转存数据库使用 SQL Sever2008 数据库软件存储了 2008—2018 年所有的软件工程领域问答文本数据。该转存数据库存储了软件知识问答交互过程中的详细信息，为软件知识图谱构建提供基础数据支持。

(2) 生成语料集。在软件知识社区 StackOverflow 中，标签 tag 代表问题所属知识领域，用户会根据所提问题的涉及的领域打 1～5 个标签。因此，借助软件知识社区 StackOverflow 的标签系统可以对海量的软件知识社区文本进行初步分类，标签所涵盖的问题数量越多，表明该标签涉及的领域受关注的程度越高，从而问题得到回答或者产生高质量答案的概率就越大。

基于此，为了获取高质量的软件知识社区文本语料集，选取问答文本的策略：首先对软件知识社区 StackOverflow 中所有标签按所包含帖子的数量进行排序，排序靠前的帖子具有较高的关注度，包含高质量软件知识的概率就越大；然后根

据标签的主题随机选择不同领域的帖子，并从所选中帖子的问题标题、问题描述、接受答案和随机选择一条评论等部分抽取对应的文本内容，作为软件知识社区文本语料集的内容；最后经过去 HTML 标签等数据预处理得到软件知识社区文本语料集。

(3) 模型训练。利用 Google 公司提供的预训练语言模型 $BERT_{Base}$ 对软件知识社区文本语料集中的 43594128 个句子(包括 2656719 个问题、5526559 个答案和评论)进行无监督训练。在训练过程中，当训练样本大小设为 32、学习率设为 0.0001、句子最大长度设为 128 时，模型的损失最低，达到最佳的训练效果。

至此，获得一个面向软件工程领域的精调预训练语言模型 SWBERT，可以为下游软件知识实体和关系抽取提供支持。

2. 软件工程领域标注数据集构建

为了获取高质量的语料文本，采用上述相同的策略构建软件工程领域标注数据集。

实体类型的预定义是实体抽取的关键基础，反映出实体的粒度和知识图谱构建的目标。相较于人物、组织、地点等通用领域的预定义实体类型，软件知识图谱属于领域知识图谱，需要结合软件工程领域的知识需求和软件知识图谱构建的目标进行更为细致的实体分类。从当前的相关研究工作看来，软件工程领域缺乏统一、完备的软件知识实体的类型定义标准，多是结合软件知识的应用目标进行预定义。我们的目标是从软件知识社区文本中抽取软件开发人员关注的特定软件知识实体，为软件开发人员提供实体为中心的软件知识服务。

因此，我们结合维基百科的软件知识分类体系和软件开发人员的知识需求，在文献[13]的基础上对软件知识实体的类型进行了扩展和细化，定义了编程语言、系统平台、软件 API、软件工具、软件开发库、软件框架、软件标准、软件开发过程 8 方面共 40 个实体类型，见表 3-2。

表 3-2 软件知识实体类型

实体类型	标签	实例	实体类型	标签	实例
面向对象语言	PLOO	Java,python,c#	科学计算软件	ToolSCS	Matlab,ATLAB
面向过程语言	PLPD	Ada,C,Fortran	集成开发工具	ToolIDE	XcodeE,Pycharm
脚本语言	PLSL	JS,Groovy,Jscript	软件开发插件	ToolSDP	JProfiler
Web 开发语言	PLML	HTML,XML	数据库软件	ToolDB	Oracle,MySQL
结构查询语言	PLSQL	SQL,GraphQL,linq	应用软件	ToolAS	Word,Excel
指令集	PlatCIS	IA-32,x86-64	视觉开发库	SLVDL	OpenCV
命令解析器	PlatCP	bash,sh,ksh	游戏开发库	SLGDL	Cocos2D

续表

实体类型	标　签	实　例	实体类型	标　签	实　例
云计算平台	PlatCCP	Hadoop,amazon-s3	移动开发库	SLMDL	Retrofit
Web 服务器	PlatCWSW	apache,nginx	软件通用库	SLSL	jQuery
操作系统	PlatCOS	Android,Ubuntu	Web 应用框架	SFWAF	jsf,Django
包/类/接口	APIP	Java. Lang	服务端框架	SFSSF	Spring 4. 2
方法/函数	APIPM	charAt(),toString()	数据格式	StanDF	. jpg,. png
数据库查询	APIDQ	LIKE,select	标准协议	StanSP	SMTP,ssl
Web API	APIWA	POST,GET	编码规范	StanCS	utf-8,gbk
系统 API	APISYS	GetMessagePos	软件设计模式	StanSDP	mvc,REST
事件驱动 API	APIOE	onClickListener	软件操作	StanSO	download
软件工程建模	SDPSEM	umlet,green-uml	软件项目部署	SDPSPD	Maven,gradle
软件测试	SDPSFT	LoadRunner	版本控制	SDPVC	VSS,CVS
Bug 异常	SDPBUG	SyntaxError	编程算法	SDPALG	Buble sort

在数据标注过程中,标注小组由 10 个具有软件工程领域背景的教师、软件开发人员、研究生、本科生组成,经过 5 轮交叉验证后,得到软件知识实体抽取任务的标注数据集,并按 7∶1∶2 的比例划分为训练集、验证集和测试集,见表 3-3。

表 3-3　数据集详细信息

信　息	训　练　集	验　证　集	测　试　集	合　计
问题	585	85	168	838
答案	1245	179	363	1787
句子	8912	1271	2568	12751
Tokens	438417	25839	51687	515943
实体	20807	2903	5917	29627

3.4.2　超参数设置

在软件知识实体抽取模型 AS-SNER 的训练过程中,输入嵌入层的预训练词向量维度设置为 768 维,上下文编码层的 BiLSTM 隐层状态的单元数设置为 200,GCN 设置为 1～3 层,采用 Categorical Cross Entropy 作为模型的损失函数,Adam 作为优化器,初始学习率设为 1e－3。同时,采用 L2 正则化和 Dropout 机制防止模型训练过拟合。模型相关超参数设置如表 3-4 所示。

表 3-4 模型相关超参数设置

参数名称	参数值
Word embedding dimension	768
BiLSTM state size	200
GCN layer	1～3
Batch size	10
Epochs	1000
Optimizer	Adam
Dropout	0.5
Learningrate	1e－3

3.4.3 对比实验结果与分析

为了验证我们提出的软件知识实体抽取模型 AS-SNER 的性能，分别选取了 CRF 模型[126]、BiLSTM-CRF 模型[127]和 BERT-BiLSTM-CRF 模型三个经典的实体抽取模型作为基线方法进行模型对比实验，实验数据采用我们构建的软件知识实体抽取标注数据集。模型对比实验的结果如表 3-5 和图 3-5 所示。

表 3-5 实体抽取结果对比

模型	*P*/%	*R*/%	*F*1/%
CRF	56.21	41.63	47.83
BiLSTM-CRF	67.74	60.15	63.72
BERT-BiLSTM-CRF	71.59	67.05	69.25
AS-SNER	**76.87**	**71.58**	**74.13**

从模型对比实验结果可知，软件知识实体抽取模型 AS-SNER 的精确率 P 值、召回率 R 值和 $F1$ 值均高于其他三个基线模型。与基于深度学习方法的 BiLSTM-CRF 模型相比较，软件知识实体抽取模型 AS-SNER 的精确率 P 值和 $F1$ 值分别提升了 9.13%和 10.41%，说明该方法基于预训练语言模型 BERT 和 GCN 能更好地学习软件知识社区文本中句子序列的领域特征和语义特征，进而缓解实体歧义问题，提升实体抽取的准确性。与基于深度学习方法的 BERT-BiLSTM-CRF 模型相比较，软件知识实体抽取模型 AS-SNER 的精确率 P 值和 $F1$ 值分别提升了 5.28%和 4.88%，说明该方法引入基于注意力权重的语义增强策略，能获取句子序列中实体的语义增强表示，有助于识别软件工程领域的常用词和因拼写错误造成的罕见词，进而缓解软件知识社区文本的实体稀疏和无法识别未登录词问题。相比较其他三个基于深度学习方法的模型，基于机器学习方法的 CRF 模型依赖人

工设定状态特征和转移特征，实体抽取的性能最差。

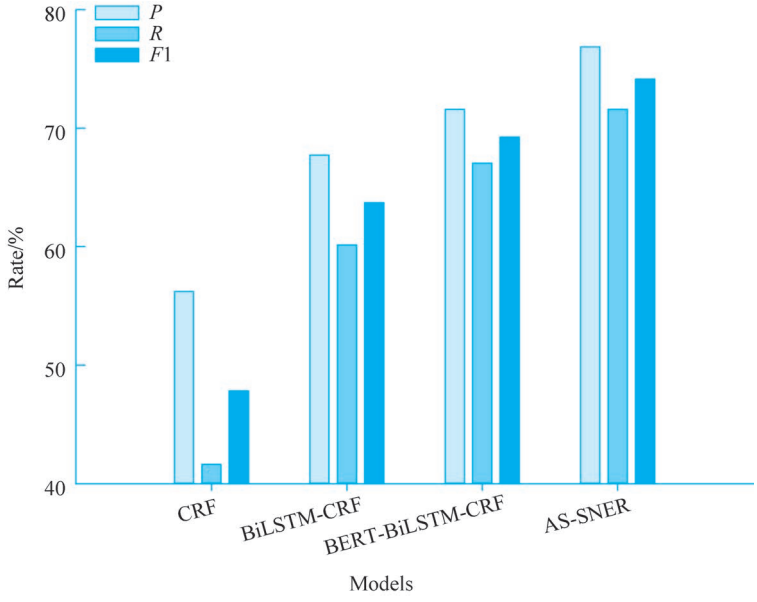

图 3-5　实体抽取结果对比

3.4.4　消融实验结果与分析

我们对软件知识实体抽取模型 AS-SNER 进行模型消融实验，主要目标是验证模型的各个组件对软件知识实体抽取的性能影响。为保证实验结果的公平性，在消融实验的过程中，各模型的相关参数保持相同的设置。

1. BERT 模型对模型性能的贡献评价

为评价预训练词向量对软件知识实体抽取任务的性能贡献，我们以 BiLSTM-CRF 模型为基准模型进行对比实验，实验结果如表 3-6 和图 3-6 所示。

表 3-6　BERT 模型对模型性能的影响

模　　型	*P*/%	*R*/%	*F*1/%
BiLSTM-CRF	67.74	60.15	63.72
BERT-BiLSTM-CRF	**71.59**	**67.05**	**69.25**

从模型对比实验结果可知，BERT-BiLSTM-CRF 模型优于基准模型 BiLSTM-CRF 模型。具体来说，加入 BERT 预训练词向量之后，模型的精确率 *P* 值提升了 3.85%，召回率 *R* 值提升了 6.9%，*F*1 值提升了 5.53%，说明基于 Transformer 的

预训练语言模型 BERT 模型具有较好的文本特征提取能力，能增强模型输入嵌入层的文本特征表示，从而提升软件知识实体抽取任务的性能。

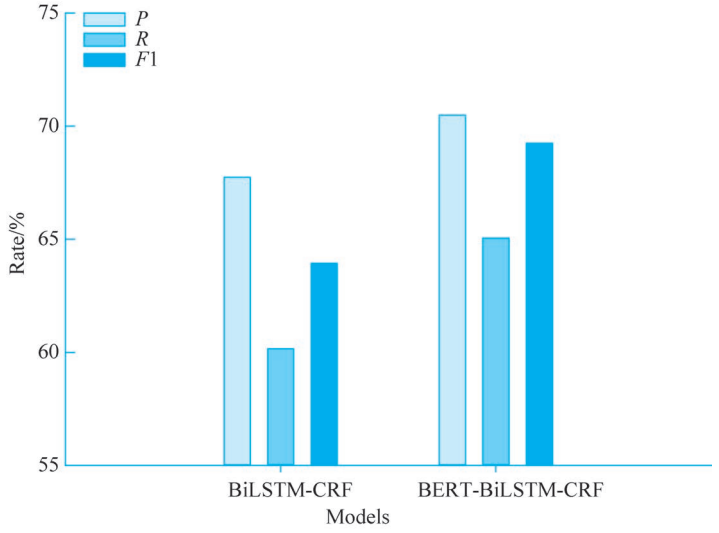

图 3-6 BERT 模型对模型性能的影响

2. 基于 BiLSTM 模型的上下文特征编码对模型性能的贡献评价

为了探索基于 BiLSTM 模型的上下文特征编码器对软件知识实体抽取任务的影响，我们选择 BERT-CRF 模型作为基准模型，对比实验了单向 LSTM 模型、BiLSTM 模型以及 BiLSTM 模型不同叠加层数的软件知识实体抽取效果。实验结果如表 3-7 和图 3-7 所示。

表 3-7 上下文特征编码对模型性能的影响

模　　型	*P*/%	*R*/%	*F*1/%
BERT-CRF	68.17	63.32	65.66
BERT-LSTM-CRF	69.04	63.54	66.18
BERT-BiLSTM-CRF($L=1$)	**71.59**	**67.05**	**69.25**
BERT-BiLSTM-CRF($L=2$)	70.23	65.38	67.72

从模型对比实验结果可知，在加入上下文编码后，相较于基准模型 BERT-CRF，模型的精确率 P 值和 $F1$ 值都有提升，说明句子序列的上下文特征提取有助于保留文本语义信息，缓解实体歧义问题。同时，BiLSTM 模型相较于单向 LSTM 模型，$F1$ 值提升了 3.07%，说明双向 LSTM 能同时捕获句子序列的上文和下文信

息，有助于句子序列的上下文特征编码。

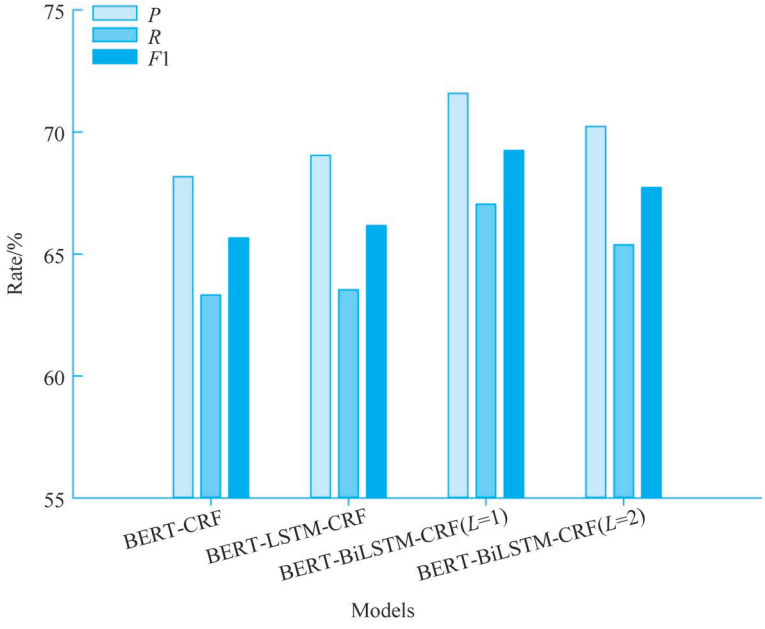

图 3-7　上下文特征编码对模型性能的影响

对 BiLSTM 模型进行堆叠可以形成深度 BiLSTM 网络，但从实验结果来看，当叠加 2 层 BiLSTM 时，模型的精确率 P 值和 $F1$ 值均有所下降，说明虽然深度 LSTM 网络具有较大的模型容量，但是模型会陷入局部最优解或过拟合问题。

3. 基于 GCN 模型的句法依存特征编码对模型性能的贡献评价

为了验证基于 GCN 模型的句法依存特征编码器对软件知识实体抽取任务的影响，我们选择 BERT-BiLSTM-CRF 模型为基线模型，对比实验了没有融合句法依存信息、融合句法依存信息情况下不同 GCN 模型叠加层数对软件知识实体抽取性能的影响，实验结果如表 3-8 和图 3-8 所示。

表 3-8　句法依存特征对模型性能的影响

模　　型	P/%	R/%	$F1$/%
BERT-BiLSTM-CRF	71.59	67.05	69.25
BERT-BiLSTM-GCN-CRF($L=1$)	73.14	67.49	70.21
BERT-BiLSTM-GCN-CRF($L=2$)	**73.77**	**69.23**	**71.43**
BERT-BiLSTM-GCN-CRF($L=3$)	71.32	68.42	69.84

从对比实验结果来看，融合句法依存特征后模型的 $F1$ 值都有所提升，说明句法依存特征有助于软件知识实体抽取任务的性能提升。同时，结果表明 GCN 的叠加层数为 2 时，模型的精确率 P 和 $F1$ 值最高，相比层数为 1 时模型的性能均得到提升；但是，当 GCN 的叠加层数为 3 时，模型的精确率 P 和 $F1$ 值均有所下降，说明 GCN 叠加层数增多会导致模型过拟合问题出现。

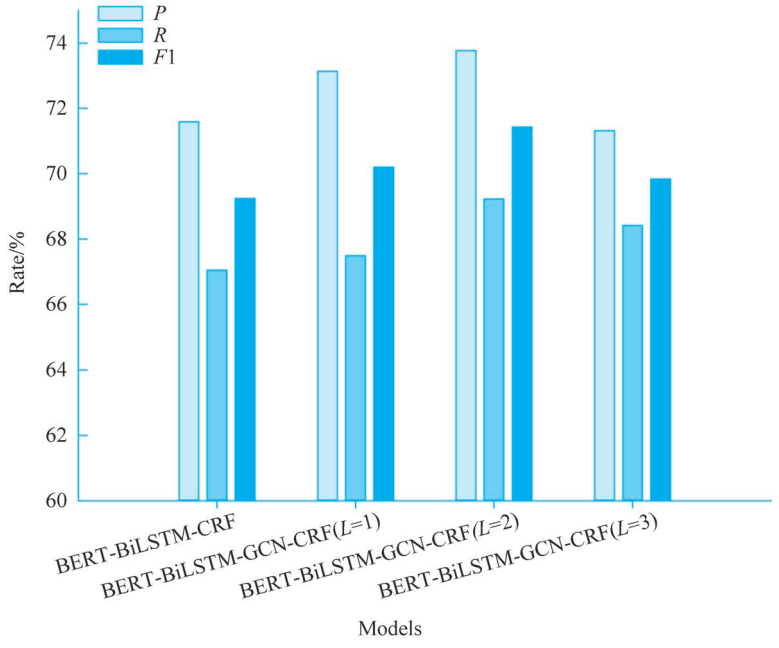

图 3-8 句法依存特征对模型性能的影响

4. 基于注意力权重的语义增强策略对模型性能的贡献评价

基于注意力权重的语义增强策略是从 BERT 预训练词向量和软件工程领域词库 SEthesaurus 中抽取相似词，并基于注意力机制对相似词的语义贡献进行权重分配，再通过加权求和得到单词的语义增强表示，从而缓解实体变体和无法识别未登录词问题。

例如，在模型训练过程中，通过基于注意力权重的语义增强策略，软件知识实体“centos”获得“rehel”“debian”“centos6”等语义相似词及其对应的语义贡献度。由图 3-9 可见，对于软件知识实体“centos”来说，实体“rehel”的语义贡献度最大，“centos6”“rehel7”等实体变体也作为辅助资源对目标实体的语义表示具有相应贡献。因此，基于注意力权重的语义增强策略能缓解软件知识社区文本存在的实体变体问题，提升模型的领域适应性。

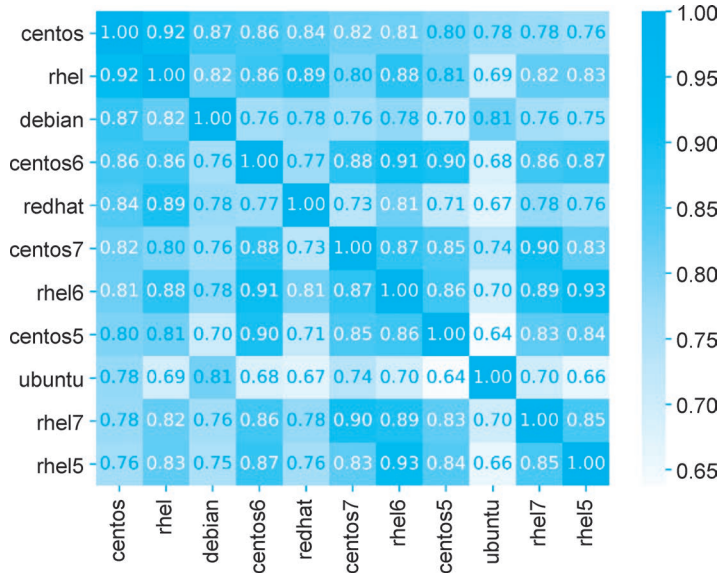

图 3-9　相似词及其语义贡献度示例

为了评价基于注意力权重的语义增强策略对软件知识实体抽取任务的性能贡献，我们选取 BiLSTM-CRF、BERT-BiLSTM-CRF、BERT-BiLSTM-GCN-CRF 等模型为基线模型进行对比实验，实验结果如表 3-9 和图 3-10 所示。

表 3-9　语义增强策略对模型性能的影响

模　　型	词特征	句法特征	语义增强	*P*/%	*R*/%	*F*1/%
BiLSTM-CRF	×	×	×	67.74	60.15	63.72
BERT-BiLSTM-CRF	√	×	×	71.59	67.05	69.25
BERT-BiLSTM-GCN-CRF	√	√	×	73.77	69.23	71.43
AS-SNER	√	√	√	**76.87**	**71.58**	**74.13**

注：若模型使用相应的特征表示，则用符号“√”表示；否则，用符号“×”表示。

从对比实验结果来看，AS-SNER 模型引入基于注意力权重的语义增强策略后，模型的综合性能得到了提升，*F*1 值较其他三个模型分别提升了 10.41%、4.88%、2.7%，说明为句子序列的单词提供领域相关的相似词，并结合注意力机制分配不同的权重，能有效增强单词的语义表示信息，进而提升软件知识实体抽取效果。

为了探索基于注意力权重的语义增强策略对无法识别未登录词问题的影响，我们对 AS-SNER 模型的实体抽取结果进一步分析。软件工程领域标注数据集的

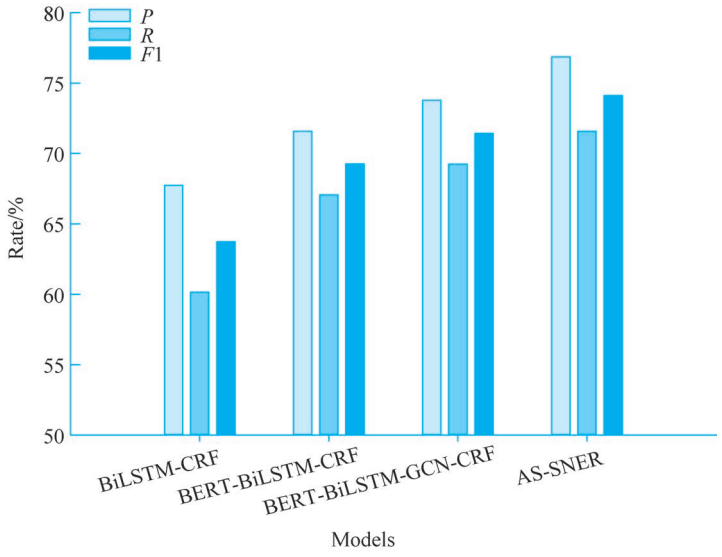

图 3-10 语义增强策略对模型性能的影响

训练集含有 20807 个软件知识实体，测试集含有 5917 个软件知识实体，其中，测试集含有 341 个未登录词。通过计算分析，各个模型对测试集中的未登录词进行识别的结果如表 3-10 所示。

表 3-10 未登录词的识别结果

模　　型	*P*/%	*R*/%	*F*1/%
BiLSTM-CRF	41.63	35.41	38.27
BERT-BiLSTM-CRF	53.45	42.39	47.28
BERT-BiLSTM-GCN-CRF	52.19	45.87	48.83
AS-SNER	**58.47**	**59.48**	**58.97**

从结果来看，引入语义增强策略的 AS-SNER 模型的召回率 *R* 值均高于其他三个未融合语义增强策略的模型，在识别未登录词上具有较好的性能。说明基于注意力权重的语义增强策略通过融合语义相似词的向量表示，能增强实体的语义表示，有助于解决软件知识社区文本存在的无法识别未登录词的问题，进而缓解实体稀疏问题。

5. 模型训练对比分析

深度学习模型的训练过程是参数不断更新的过程，为了进一步了解模型的训练情况，选取了 BiLSTM-CRF、BERT-BiLSTM-CRF、BERT-BiLSTM-GCN-CRF

和 AttenSy-SNER 模型训练前 100 轮 Epoch 的过程数据进行了对比分析。各模型的 $F1$ 值和 Epoch 的关系如图 3-11 所示。

从如图 3-11 可知，没有利用 BERT 模型的 BiLSTM-CRF 模型，$F1$ 值从较低的初始值连续不断提升；其他三个利用 BERT 模型生成动态词向量的模型，$F1$ 值获得较高的初始值，并持续保持较高的水平。这也反映出利用 BERT 模型作为输入嵌入层的特征编码器，能有效提取软件知识社区文本特征，对软件知识实体抽取任务的性能具有重要贡献。

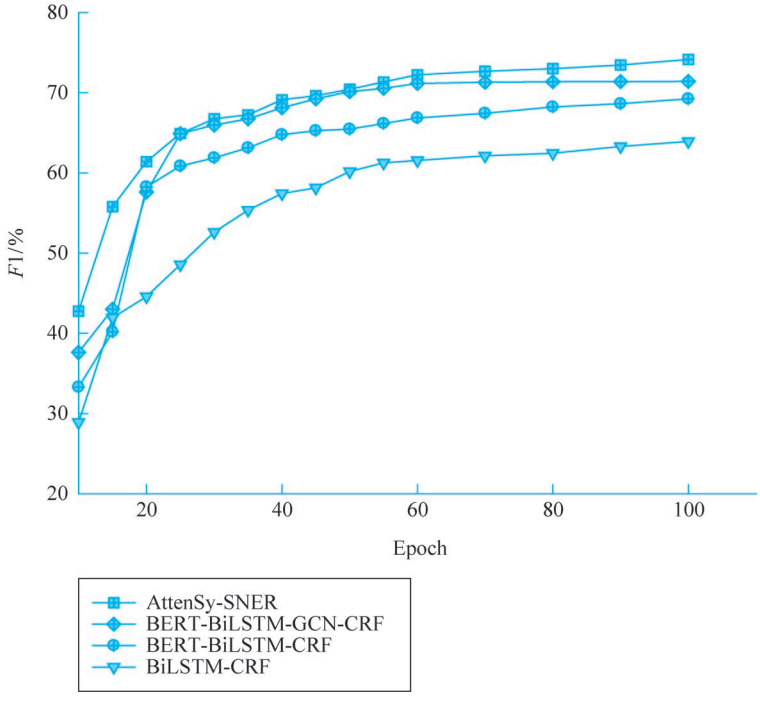

图 3-11　$F1$ 值和 Epoch 的关系

相比较其他模型，融入上下文语义特征、句法依存特征和领域实体语义增强的 AttenSy-SNER 模型，在初始阶段就能得到最高的 $F1$ 值，在第 40 轮 Epoch 时损失函数开始收敛，并持续保持最佳的 $F1$ 值状态。

3.5　本章小结

我们提出了一种基于多特征融合和语义增强的软件知识实体抽取方法，并以软件知识社区 StackOverflow 为例进行了实验与分析。实验结果表明，利用

BiLSTM 模型和 GCN 模型对句子序列的上下文特征和句法依存特征进行融合，能带来模型性能的提升。同时，通过基于注意力权重的实体语义增强策略能有效增强单词的语义表示信息，进而提升软件知识实体抽取效果。针对标注数据集缺乏的问题，基于软件知识社区 StackOverflow 的问答文本构建了涵盖 8 方面 40 个实体类型的软件工程领域标注数据集。