知识增强预训练语言模型综述

王楠[1)]

[1)] (华中科技大学，湖北 武汉 430074)

摘 要 近年来，知识图谱越来越受到人们的关注，知识图谱旨在提供一种复杂但灵活的知识的显式建模。许多专家认为，这种新型的知识表示对于预训练模型是一种良好的补充。因此，本文将知识增强预训练语言模型重点转向知识图谱融合预训练语言模型，搜集最新相关研究成果，从模型的角度入手，按照知识与预训练模型融合的方式进行分类，并选取每类具有代表性的工作具体介绍，重点分析模型相比其他预训练语言模型的优势，研究知识对于预训练语言模型性能提升的增强作用，并评估对比模型，最后分析当前预训练语言模型发展过程中所面临的问题和挑战，对领域发展前景进行展望。

关键词 知识图谱；预训练语言模型；注意力机制；知识图谱嵌入；深度学习

A Review of Knowledge Enhancement Pre-training Language Models

Wang Nan[1)]

[1)](Huazhong University of Science and Technology, HuBei Wuhan 430074)

Abstract In recent years, knowledge graph has attracted more and more attention. Knowledge graph aims to provide a complex but flexible explicit modeling of knowledge. Many experts believe that this new type of knowledge representation is a good complement to the pre-training model. Therefore, this article knowledge to enhance the training of language model emphasis to training language model, knowledge map fusion to collect the latest relevant research results, proceed from the Angle of model, according to the knowledge and the training model fusion method to classify, concrete is introduced, and the selection of each type of representative work focuses on analyzing model compared with other advantages, in the process of the training of language model This paper studies the enhancement effect of knowledge on the performance improvement of pre-trained language models, evaluates the comparative models, and finally analyzes the problems and challenges faced in the development process of pre-trained language models, and prospects the development of the field.

Key words knowledge graph; pre-trained language models; attention; knowledge graph embedding; deep neural network

1 引言

自然语言处理（Natural Language Process- ing，NLP）是人工智能和语言学领域的分支学科，近年来，NLP 的发展取得了巨大进步，任务划分也更加详尽，其主要下游任务，如关系抽取、文本分类、对话生成、机器翻译和共指消解等均离不开预训练技术。预训练技术使用大规模无标注的文本语料库对深层网络结构进行训练，从而得到一组模型参数，这种深层网络结构通常被称为“预训练模型”。将训练得到的模型参数运用于后续特定的下游任务，可避免从零开始进行再训练的繁琐过程。预训练模型的优点在于训练代价较小，配合下游任务能够达到更好的收敛效果，可以较好地提升模型的性能。

自 BERT 模型被提出以来，各种预训练语言模型层出不穷。然而，非知识增强的预训练语言模型通常采用浅层网络结构，只进行简单的学习，无法理解更高层次的文本概念，如语义角色、指代等，且在一些涉及逻辑推理和认知的任务上力有不逮；而知识增强的预训练语言模型能够根据不同的应用目的而加强不同的模块，使模型具备更加专业的业务能力。因此，面对智能要求和专业化程度更高

的深层次 NLP 任务，需要探索并分析将特定种类的外部知识嵌入到特定用途的预训练语言模型中。

目前，已有部分研究人员尝试将外部知识嵌入到预训练语言模型中，评估其在特定下游任务中的性能表现。NLP 的外部知识含义广泛，涵盖了语言学知识、语义知识、常识知识、事实知识以及领域知识等。知识丰富的预训练语言模型通常从通用的大规模文本语料库中学习通用的语言表示，以灵活应对多种下游任务并进行相应的微调。基于知识增强对预训练语言模型进行扩展，使得预训练技术进入了一个新的发展阶段。

近年来，知识图谱越来越受到人们的关注，知识图谱旨在提供一种复杂但灵活的知识的显式建模。许多专家认为，这种新型的知识表示对于预训练模型是一种良好的补充。本文将重点转向知识图谱融合预训练语言模型，搜集最新相关研究成果，分析此类模型相比其他预训练语言模型的优势，研究知识对于预训练语言模型性能提升的增强作用，并评估对比模型，最后进行总结并对未来的研究方向进行展望。

2 原理和优势

2.1 方法优势

我们期望，预训练模型能够在需要特定知识或常识的任务中表现得更好。例如，在机器阅读理解中，给定一句《哈利波特》小说中的句子：“Harry Potter points his wand at Lord Voldmort.”，如果要让模型理解这句话，比如哈利波特、伏地魔之间的时空关系，模型就需要知道一些《哈利波特》中的先验知识，让模型对文本中某些实体的有更好的理解，而不仅仅是把它们当作一个称谓（指代）。在文本生成中，如果要让模型用“河、鱼、网、捉”几个词造句，至少应当保证生成的句子符合正常的逻辑关系，比如“人在河里用网捉鱼”，而不是“鱼在捉网”等，这里面就涉及到一些常识。

尽管这些常识已经通过大量的文本预训练或多或少隐式存储在了模型中，但这些知识并不足以让预训练模型处理当下的任务，所以才需要进行知识的注入和增强。另一方面，得到了知识的增强之后，预训练模型可以表现得更像是领域专家，或更像是生活中插科打诨的人。比如在和对话机器人聊天，说到：“翻译翻译，什么叫惊喜。”是更希望它把“惊喜”翻译成英文，还是更想让它玩汤师爷的梗。

2.2 面临的问题

将知识图谱信息注入到预训练模型中，常见的三个问题是：

1）结构化信息的非结构化；

2）异构特征空间的对齐；

3）知识噪声的解决。

第一个问题是结构化信息的非结构化。相对于自然语言文本这种非结构化的信息，知识图谱在形式上可以看作是三元组的列表，或者是一张有向图，这样的形式蕴含着比自然语言文本更多的信息，即结构化的信息。如何将这样的信息与擅长提取非结构化文本信息的预训练模型结合起来，是开展工作的第一步。

第二个问题是空间的对齐。由于 token 部分的特征与知识图谱的 embedding 是由两种不同的方法得到的，二者所处的特征空间是不完全相同的。针对这类问题，最简单的方法是学习一个线性变换去做向量的对齐。

第三个问题是知识噪声。如果无法进行良好的融合，所融入的知识图谱信息不仅不会提升性能，反而还会降低预训练模型的效果。在 embedding 层面，知识信息可能会干扰注意力机制的运算，使得当前的推理与预训练模型参数之间发生矛盾；对于在输入文本上的融合，知识噪声可能会破坏原有句子的结构和表达，从而影响模型的理解和生成。

下面介绍的相关工作都是基于这三个问题提出的思考。

2.3 相关工作

用知识图谱增强预训练模型，存在多种角度解决该问题的方法。

从主客关系上看，部分通过知识图谱增强预训练模型，以实现更好的知识图谱构建，另一部分通过知识图谱将知识引入预训练模型，来提升其在 NLP 本身的下游任务上的效果。

从知识图谱与预训练模型的增强方式来看，部分通过线性变化、注意力机制等在知识实体与关系的嵌入式表达上进行知识融合，部分直接使用实体描述，设计特定的预训练模型输入。在面向的任务上，除去 GLUE 等进行通用自然语言理解测试的基准之外，还包括近期热门的知识增强的开放领域问答、常识文本生成等等。

由于相关工作过于庞杂，本文从模型的角度入手，按照知识与预训练模型融合的方式进行分类，仅选取具有代表性的工作进行重点分析。倘若粗略划分，知识融合的方法可简单分成 embedding 层面上的融合，token 层面的融合，以及知识图谱与预训练模型的共同学习三类。

3 研究进展

本章分别选取三类方法即隐式融合—使用 embedding 在模型内部融合、显示融合—不改变模型结构的融合方式，以及知识图谱与预训练模型的共同学习中的典型模型进行具体介绍，并在最后一节进行对比分析。

3.1 ERNIE

3.1.1 研究背景

为了克服将知识整合进语言表征模型的两个挑战：结构化知识编码和异构信息融合，Zhang[1] 等人提出基于信息实体的增强语言表示(ERNIE)，它在大规模文本语料库和 KG 上预训练了语言表示模型：

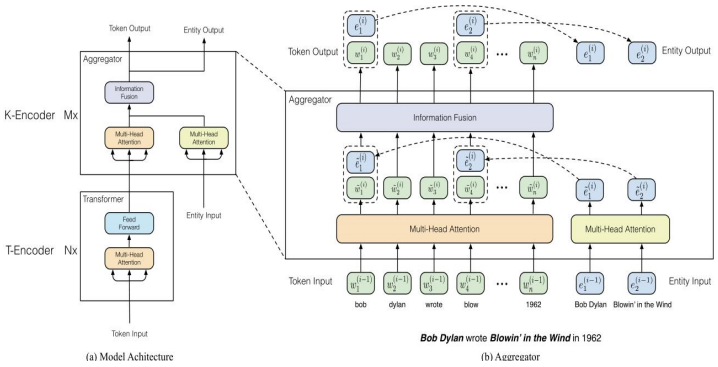
1）为了提取和编码知识信息，首先识别文本中提及的命名实体，然后将这些提及到的实体与它们在 KG 中的相应实体对齐。没有直接使用 KG 中基于图的事实，而是使用诸如 TransE 之类的知识嵌入算法对 KG 的图结构进行编码，然后将信息实体嵌入作为 ERNIE 的输入。基于文本和 KG 之间的对齐，ERNIE 将知识模块中的实体表示集成到语义模块的底层。

2）与 BERT 类似，采用掩码语言模型和下一句预测作为预训练目标。此外，为了更好地融合文本和知识特征，设计了一个新的预训练目标，通过随机 mask 输入文本中的一些命名实体对齐并要求模型从 KG 中选择合适的实体来完成对齐。与现有的仅利用局部上下文来预测标记的预训练语言表示模型不同，目标是要求模型聚合上下文和知识事实以预测 token 和实体，并形成知识增强的语言表示模型。

3.1.2 设计与实现

1）符号说明

将标记序列表示为 $\{w_1, ..., w_n\}$，其中n是 token 序列的长度，同时，将与给定 token 对齐的实体序列表示为 $\{e_1, ..., e_m\}$，其中m是实体序列的长度。需要注意的是，在大多数情况下m不等于n，因为并非每个 token 都可以与 KG 中的实体对齐。此外，将包含所有标记的整个词汇表表示V，将包含 KG 中所有实体的实体列表表示为E。如果标记$w \in V$有相应的实体$e \in E$，则它们的对齐定义为$f(w) = e$。在本文中，将实体与其命名实体短语中的第一个标记对齐，如图 1 所示。图 1 左边是 ERNIE 的架构。右边是 token 和 entity 的输入相互集成的聚合器。信息融合层有两种输入:一种是 token em- bedding，另一种是 token embedding 和 entity embedding 的串联。信息融合后，它为下一层输出新的 token embedding 和 entity embedding。



（a）模型架构　　（b）聚合器

图 1 模型架构和聚合器图

2）模型架构

如上图 1 所示，ERNIE 的整个模型架构由两个堆叠模块组成：

底层文本编码器 (T-Encoder) 负责从输入标记中捕获基本的词汇和句法信息；

上层知识编码器 (K-Encoder) 负责将额外的面向 token 的知识信息整合到来自底层的文本信息中，以便将 token 和实体的异构信息表示到一个统一的特征空间中。此外，将 T-Encoder 层数表示为 N，将 K-Encoder 层数表示为M。从文章后面实验可以看出，作者其实是让$M = N$。

具体来说，给定一个 token 序列 $\{w_1, ..., w_n\}$，及其对应的实体序列 $\{e_1, ..., e_m\}$，文本编码器首先对每个 token 的 token embedding, segment em- bedding, positional embedding 求和以计算其 input embedding，然后利用如下公式计算 $\{\boldsymbol{w}_1, ..., \boldsymbol{w}_n\}$ (加粗表示向量或者矩阵)，其中，$T - En\text{coder}(\cdot)$是一个多层双向 Transfomer，与 BERT 中的实现相同。

$$\{\boldsymbol{w}_1, ..., \boldsymbol{w}_n\} = T - \text{Encoder}(\{w_1, ..., w_n\})$$

在计算 $\{\boldsymbol{w}_1, ..., \boldsymbol{w}_n\}$，ERNIE 采用知识编码器 $K-Encoder$将知识信息注入语言表示。具体而言，用实体嵌入 $\{\boldsymbol{e}_1, ..., \boldsymbol{e}_m\}$ 表示实体 $\{e_1, ..., e_m\}$，该篇文章通过高效的知识嵌入模型 TransE 进行预训练得到实体 embedding，然后，将两者共同输入到$K-Encoder$，目的是融合异构信息和计算最终的 output embedding。

$$\{\boldsymbol{w}_1^o, ..., \boldsymbol{w}_n^o\}, \{\boldsymbol{e}_1^o, ..., \boldsymbol{e}_n^o\} = K-\text{Encoder}\left(\{\boldsymbol{w}_1, ..., \boldsymbol{w}_n\}, \{\boldsymbol{e}_1, ..., \boldsymbol{e}_m\}\right)$$

$\{\boldsymbol{w}_1^o, ..., \boldsymbol{w}_n^o\}$和$\{\boldsymbol{e}_1^o, ..., \boldsymbol{e}_n^o\}$ 将作为特征用于特定的任务。

其中，$K-Encoder$ 由多个堆叠的聚合器组成。在第 i 个聚合器中，将来自前面的聚合器的 input token embedding 和 entity embedding 分别输入两个多头自注意力（MH-ATTs），且采用信息融合层对 token 和实体序列进行相互集成，并计算每个 token 和实体的 output embedding。对于一个 token w_j及其对齐的实体$e_k = f(w_j)$，信息融合过程如下式：

$$h_j = \sigma\left(\tilde{\boldsymbol{W}}_t^{(i)}\tilde{\boldsymbol{w}}_j^{(i)} + \tilde{\boldsymbol{W}}_e^{(i)}\tilde{\boldsymbol{e}}_k^{(i)} + \tilde{\boldsymbol{b}}^{(i)}\right)$$
$$w_j^{(i)} = \sigma\left(\boldsymbol{W}_t^{(i)}\boldsymbol{h}_j + \boldsymbol{b}_t^{(i)}\right)$$
$$e_k^{(i)} = \sigma\left(\boldsymbol{W}_e^{(i)}\boldsymbol{h}_j + \boldsymbol{b}_e^{(i)}\right)$$

其中，中间隐藏状态$\boldsymbol{h}_j$集成了 token 和实体信息。$\sigma(\cdot)$是非线性激活函数，通常是 GELU 函数。对于句子中的某些 token，它在 KG 中没有对应实体，信息融合层以相似方式计算 output embedding。由顶层聚合器计算的 token 和实体的 output embedding 将用作$K-Encoder$的最终 output embedding。

3）知识注入预训练

为了将实体知识注入到语言表示模型中，文章提出了一个新的预训练任务，随机 mask 一些 token-entity 对齐，然后要求模型基于对齐的 token 来预测所有相应的实体。由于任务类似于训练去噪自编码器，将此过程称为去噪实体自编码器(dEA)。考虑到 softmax 层的 ϵ 非常大，因此只需要根据给定的实体序列而不是 KG 中的所有实体来预测实体。给定 token 序列 $\{w_1, ..., w_n\}$ 及其对应的实体序列 $\{e_1, ..., e_m\}$，定义 token w_i的对齐实体分布如下：

$$p(e_j|w_i) = \exp(linear)/(\sum_{k=1}^{m}(linear(w_i^o)\cdot e_k)$$

其中$linear(\cdot)$是一个线性层。计算 dEA 的交叉熵损失函数会用到上述公式。

考虑到 token-entity 对齐存在一些错误，对 dEA 执行以下操作：

1.对于给定的 token-entity 对齐，以 5 %的概率用另一个随机实体替换该实体，旨在训练模型能够纠正 token 与错误实体对齐的错误；

2.以 15 %的概率 mask 掉 token-entity 对齐，旨在训练模型以纠正实体对齐系统未提取所有现有对齐的错误；

3.以 80 %的概率保持 token-entity 对齐不变，旨在鼓励模型将实体信息整合到 token 表示中，以取得更好的语言理解。

与 BERT 类似，ERNIE 也采用掩码语言模型(MLM) 和下一句预测 (NSP) 作为预训练任务，使 ERNIE 能够从文本中的标记中捕获词汇和句法信息，整体预训练损失是 dEA、MLM 和 NSP 损失的总和。

4）针对特定任务的微调

如图 2 所示，对于各种常见的 NLP 任务，ERNIE 可以采用类似于 BERT 的微调过程。可以将第一个 token 的最终输出嵌入，它对应于特殊的[CLS]标记，作为特定任务的输入序列的表示。对于一些知识驱动的任务（例如，关系分类和实体类型），文章设计专门的微调过程：

对于关系分类，该任务需要系统根据上下文对给定实体对的关系标签进行分类。不同于最直接微调 ERNIE 的方法，将池化层应用于给定实体提及的最终输出嵌入，并用它们的提及嵌入的串联来表示给定实体对以进行分类，文章设计了另一种方法，修改输入 token 序列，添加两个标记 token 来突出实体提及。这些额外的标记 token 在传统的关系分类模型中扮演着类似于位置嵌入的角色，然后，依然采用[CLS]token 嵌入进行分类。值得注意的是，分别为头部实体和尾部实体设计了不同的标记[HD]和[TL]。

实体类型的特定微调过程是关系分类的简化版本。由于以前的识别模型充分利用了上下文嵌入和实体提及嵌入，文章认为修改后的带有提及标记 token [ENT]的输入序列可以引导 ERNIE 将上下文信息和实体提及信息仔细结合起来。

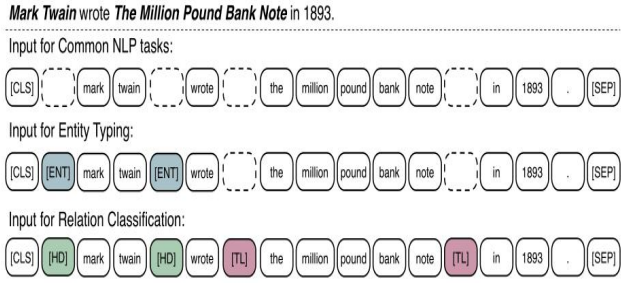


图 2 修改特定任务的输入序列

3.2 K-BERT

3.2.1 研究背景

BERT 是基于大规模开放语料的预训练模型，对于下游任务，只需微调就可吸收专业领域知识。但是由于预训练和微调之间存在领域知识差异，因而在领域知识驱动型任务上，BERT 无法取得满意的表现。

一种解决办法就是基于领域知识的语料库进行预训练，但是这样做耗时耗力，对大多数用户是不能承受的。

因此认为引入知识图谱来使得模型成为领域专家是一个很好的解决方案。但这种方法面临两个挑战，也是前述提到的异构嵌入空间和知识噪声，于是 Liu 等人[2]提出了 K-BERT 模型，K-BERT 可以加载任意 BERT 模型，然后很容易植入领域知识，而不需要再进行预训练。

3.2.2 设计与实现

1）符号说明

$s = \{w_0, w_1, w_2, ..., w_n\}$ 表示一句话， n表示 token 序列长度，对于英文是词级别的，对于中文是字级别的；

$\mathbb{V}$：字典，所有的 $w_i \in \mathbb{V}$；

$\mathbb{K}$：知识图谱，是所有三元组 ε 的集合；

$\varepsilon = (w_i, r_j, w_k)$：一个三元组， $\varepsilon \in \mathbb{K}$。

2）模型架构

K-BERT 的模型结构如下图 3 所示，模型包括四个模块，分别是 Knowlegde layer、Embedding layer、seeing layer 和 Mask-Transformer Encoder。对于输入的句子，知识层首先从 KG 中注入相关的三元组，将原句子转化为富含知识的句子树，然后句子树被同时送入嵌入层和可见层然后转换成 token 级嵌入表示和可见矩阵，使用可见矩阵来控制每个 token 的可见区域，防止由于注入过多的知识而改变原句的意思。

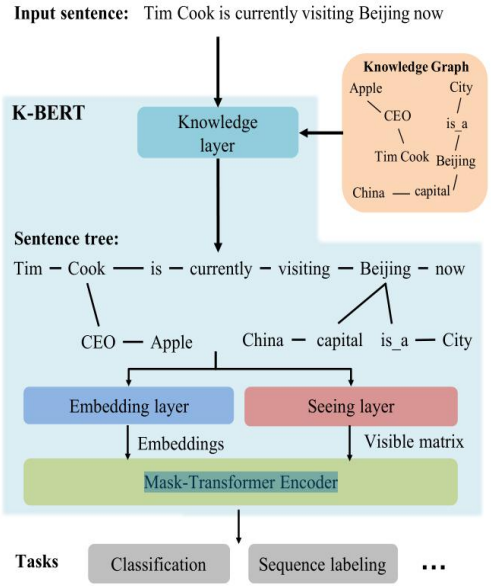


图 3 K-BERT 模型结构

Knowlegde layer

Knowledge layer(KL)的作用是 1.将知识图谱注入到句子中；2.完成句子树的转换。

给定知识图谱 $\mathbb{K}$，输入一句话 $s = \{w_0, w_1, w_2, ..., w_n\}$，经 KL 层后输出句子树 $t = \{w_0, w_1, ..., w_i\{(r_{i0}, w_{i0}), ..., (r_{ik}, w_{ik})\}, ..., w_n\}$，该过程分为两步：知识查询 (K-Query)和知识注入 (K-Inject)。K-Query 步骤针对句子中的所有实体都查询一遍 KG，K-Inject 注入知识生成句子树的过程如下图 4 所示。

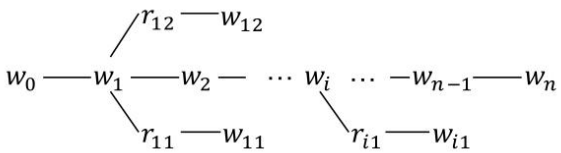


图 4 句子树结构

Embedding layer

Embedding layer(EL)的作用是将句子树转换成嵌入表达。和 BERT 一样，K-BERT 的嵌入表示包括三部分：token embedding、position embedding 和 segment embedding。关键要点是如何将句子树转换成一个序列，同时保留它的结构信息。

K-BERT 和 BERT token 嵌入的区别在于，在进行嵌入操作之前，句子树中的 token 需要重新排列。在文章的重新排列策略中，分支中的 token 被插入到相应节点之后，而随后的 token 向后移动，如图

5 所示。这个过程虽然简单，但却使句子难以读懂，并丢失了正确的结构信息。幸运的是，它可以通过 soft-position 和可见矩阵来解决。

对于 BERT 来说，如果没有位置嵌入，它就相当于一个词袋模型，导致缺乏结构信息(即 token 的顺序)。BERT 的输入句子的所有结构信息都包含在位置嵌入中，这允许将缺失的结构信息添加回不可读的重新排列的句子中。Soft-position embedding 的位置编码示例见图 5，虽然解决了计算 transfomer 编码器自注意评分问题时，[is]是在[Cook]的下一个位置的等效值，但另一个问题是：[is]和[CEO]的位号都是 3，这使得它们在计算自我注意时位置接近，但实际上并没有联系。这个问题的解决方案是 Mask-Self-Attention，在后续阐述。

对于 segment embedding，K-BERT 和 BERT 采用同样的方法。

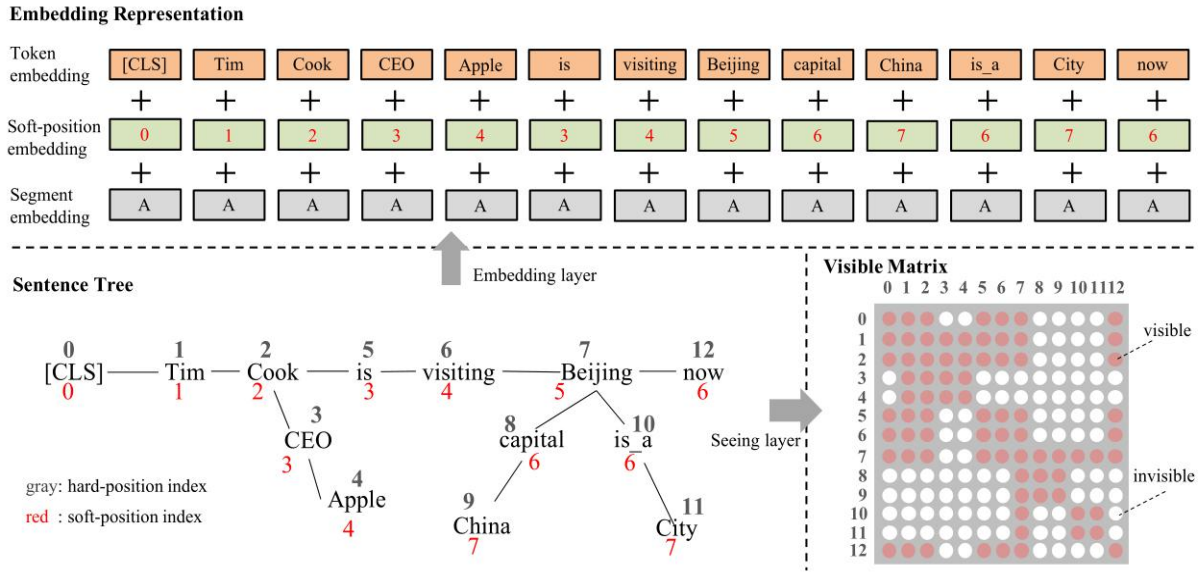


图 5 将句子树转换为嵌入表示和可见矩阵的过程

Seeing layer

Seeing layer 是 K-BERT 和 BERT 的最大区别，也是该方法有效的关键。

前文已经讲了引入 KG 可能造成 KN 问题。例如，[China]只修改了[Beijing]，与[Apple]没有任何关系，因此，[Apple]的表示不应受到[China]的影响。另一方面，用于分类的[CLS]标签不应该绕过[Cook]来获取[Apple]的信息，因为这样会带来语义变化的风险。为了防止这种情况发生于是 Seeing layer 层的作用就是通过一个可见矩阵来限制词与词之间的联系，这样[Apple]和[China]， [CLS]和[Apple]等都彼此都不可见。

可见矩阵M的定义如下：

$$M_{ij} = \begin{cases} 0 & w_i \ominus w_j \\ -\infty & w_i \oslash w_j \end{cases}$$

$w_i \ominus w_j$ 表示两个词在同一支路上，$w_i \oslash w_j$ 表示两个词不在同一支路上，i和j是 hard-position index。关于 soft-position index 和 hard-position index，见图 5。

Mask-Transformer Encoder

因为 BERT 中 Transformer 编码器不能接收可见矩阵作为输入，因此作者对其进行了改造，称为 Mask-Transformer。所谓 Mask-Transformer，其实就是 mask-self-attention 块的堆叠。

符号同 BERT，L - mask-self-attention 块的个数，H - hidden size，A - mask-self-attention header 的个数。Mask-Self-Attention 的数学定义如下式：

$$\begin{aligned} Q^{i+1}, K^{i+1}, V^{i+1} &= h^i W_q, h^i W_k, h^i W_v \\ S^{i+1} &= \text{softmax}\left(\frac{Q^{i+1} K^{i+1\top} + M}{\sqrt{d_k}}\right) \\ h^{i+1} &= S^{i+1} V^{i+1} \end{aligned}$$

其中，W_q, W_k, W_v 是需要学习的模型参数，h^i 为第 i 个 mask-self-attention 块的隐藏状态，d_k 为缩放因子，M为可见矩阵。如果 w_k 对 w_j 不可见，则 $M_{jk} = -\infty$，从而注意力 $S^{i+1} = 0$。

Mask-Transformer 的图如下图 6 所示。从图中可以看到，$h^i_{[\text{Apple}]}$对$h^{i+1}_{[CLS]}$是不可见的。但是 $h^{i+1}_{[Apple]}$是可以间接影响到 $h^{i+1}_{[CLS]}$ 的。

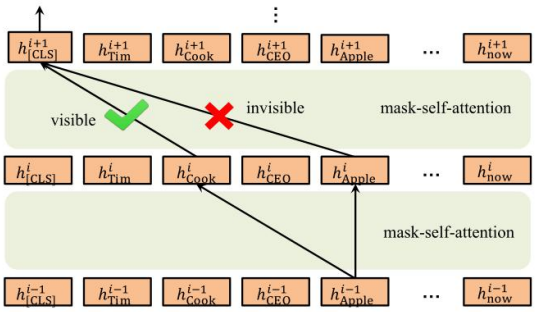


图 6 Mask-Transformer 说明图

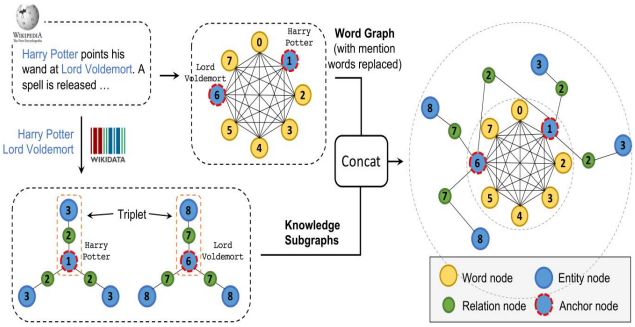


图 7 WK 图构造说明

3.3 CoLACK

3.3.1 研究背景

Sun 等人[3]在这篇文章里主要想做两件事：

1）在预训练语言模型时也学习一套知识表示，以在需要知识的下游文本任务上表现更好，例如实体链接、关系抽取等；（由于实体数量通常很多，之前的一些工作大多使用 TransE 等算法预先训练好实体表示然后融入到 PLM 中去，之后不再随 PLM 一起更新）

2）在加入实体的同时也加入它的上下文，允许模型在不同语境下关注实体的不同邻居，同时学习文本和知识的上下文表示。（之前的模型只完成了文本的语境化而没有完成知识的语境化）

3.3.2 设计与实现

1）构建 word-knowledge graph

如何将单词图和知识子图整合到统一的 WK 图中。将一个句子分词成 token 序列，并将它们完全连接成一个 word 图。然后识别句子中的提及，并使用实体链接器在某一 KG 中找到相应的实体，然后，提到的节点被它们的链接实体替换，这些链接实体被称为锚节点。通过这种替换，鼓励模型映射注入的实体，并在向量空间中提到彼此相邻的 words。以锚节点$\{e_i\}_i$为中心，提取其知识上下文形成子图，子图中的关系也转化为图节点。然后将提取的子图和 word 图与锚节点连接，得到 WK 图。图 7 显示了构建 WK 图的过程。在实践中，对于每个锚节点，随机选择最多 15 个相邻关系和实体来构造一个注入到 WK 图中的子图，且只考虑锚节点为头部（对象）而不是尾部（对象）的三元组。在 WK 图中，实体是唯一的，但关系允许重复。

2）模型架构

然后将构建的 WK 图输入 Transformer 编码器。修改了普通 Transformer 的嵌入层和编码器层，以适应以 WK 图的形式输入。

Embedding Layer

输入嵌入是 token 嵌入、类型嵌入和位置嵌入的总和。对于 token 嵌入，分别为单词、实体和关系维护了三个查找表。对于单词嵌入，遵循 RoBERTa 的方法，使用字节对编码（BPE）将序列转换为子单词单元，以处理大词汇量，相比之下，直接学习每个独特的实体和关系的嵌入，就像普通知识嵌入方法一样。通过将相同维数的字嵌入、实体嵌入和关系嵌入进行连接，得到 token 嵌入。有不同类型的节点，所以 WK 图是异构的。为了处理这个问题，简单地使用类型嵌入来表示节点类型，例如字、实体和关系。对于位置嵌入，需要为每个注入的实体和关系分配一个位置索引，受 K-BERT 的启发，采用了 soft-position 索引，它允许重复的位置索引，并且保持在同一的三元组中的 token 连续。图 8 展示了一个如何为图节点分配位置索引的直观示例。

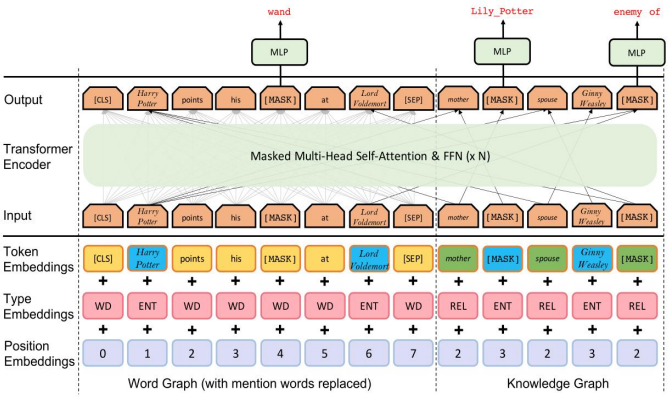


图 8　CoLAKE 整体架构

Masked Transformer Encoder

利用掩蔽多头自注意力来控制信息流，以反映WK图的结构。给定图节点的表示形式$\mathbf{X} \in \mathbb{R}^{n \times d}$，设置掩码矩阵：

$$\mathbf{M}_{ij} = \begin{cases} 0 & \text{如果 } x_i \text{和} x_j \text{是连接的,} \\ -inf & \text{如果 } x_i \text{ 和 } x_j \text{ 没有连接.} \end{cases}$$

使用masked Transformer encoder，每个节点在每一层只能从它的1跳邻居收集信息。masked Transformer encoder的工作原理类似于GAT。

3）预训练目标

掩蔽语言模型(MLM)的目标是随机从输入中mask一些tokens，并训练模型根据上下文预测被mask的tokens的原始词表id，文章将MLM从word序列扩展到WK图。

同样的，随机屏蔽15%的图节点，由于不同类型的节点被mask，鼓励CoLAKE学习不同方面的能力：

1.Masking word nodes，类似于传统的MLM，不同的是，CoLAKE不仅可以根据上下文单词，还可以根据WK图中的实体和关系预测屏蔽词

2.Masking entity nodes，如果掩蔽实体是一个锚节点，目标是根据其上下文预测锚节点，这有助于对齐语言和知识的表示空间。如果掩体实体不是锚节点，则MLM目标与基于语义匹配的KE方法类似，如ConvE等，这使CoLAKE能够学习大量的实体嵌入。

3.Masking relation nodes，如果被掩蔽的关系是在两个唯一的锚节点之间，目标类似于远程监督关系提取，这需要模型对文本中提到的两个实体之间的关系进行分类。另外，目标是预测它的两个相邻实体之间的关系，这与传统的KE方法类似。

3.4 KEPLER

该模型是知识图谱与预训练模型的共同学习类型的方法，已在论文研讨时介绍过，这里便不再阐述。

3.5 评估

隐式融合是比较直接的embedding融合方法，该类方法基于一些KGE，使用最多的是TransE算法获得知识图谱中的实体与关系的embedding，并为这些embedding修改预训练模型结构，以便将二者进行结合。根据将embedding与预训练模型结合的方式，可以粗略分为基于projection的结合方法和基于attention的结合方法。

ERNIE属于隐式融合方法，尽管embedding的方式比较直观，且属于典型的深度学习风格，但该类方法仍然存在以下几类问题：

1. 知识的融合效果受knowledge embedding学习限制；
2. 简单的变换未必能将知识空间与文本空间对齐；
3. 该结构需要对预训练模型重新进行预训练，即使通用的KG+PTM预训练是可行的，针对特定领域的预训练或微调同样存在困难。

与基于embedding的结合相比，另一种思路更为直接：由于知识图谱本身就是借助自然语言表达的，直接把实体和关系经过某些变换后，以token的形式输入到预训练模型中，这样就不太需要再去为每个实体学习embedding，更不需要考虑两种特征空间的聚合。

一些相关的工作都是在这种思路的基础上进行的，这类工作大概可以分成两类，训练时的知识增强以及推理时的知识增强。

K-BERT和CoLAKE即属于此类方法，不同于K-BERT意图将输入文本扩展成一棵树，CoLAKE的核心思路则是把输入文本通过知识图谱扩展成一张图。又因为构建起图之后，该图也是需要送入BERT等模型中去，因而CoLAKE面临着和K-BERT同样的设计问题。

CoLAKE与K-BERT的主要区别有三处：

1. segment embedding中细化了很多细节；
2. 该图的mask矩阵是基于图的拓扑关系构建，相当于是邻接矩阵，而不是根据避免知识噪声的原则构建的；
3. 该论文更为重视预训练过程，通过之前所说的扩展MLM的方法完成了对模型更好的训练。

CoLAKE的另外一个特色在于，通过这种方式得到的知识图谱中实体与关系的embedding，比TransE等传统方法的效果还要好，这一点也在一些知识图谱补全实验中得到验证。

除去上述两部分的工作之外，还有一类工作进行了额外的尝试。该类工作并不满足于单纯地将知识图谱增强于预训练模型，而是将知识图谱表示学习和面向知识的自然语言理解一起处理，希望通过这种方式可以在这两个领域内都得到模型的提升。

KEPLER 就是此类方法的一个典型模型。

4 总结

目前，知识增强的预训练语言模型在 NLP 任务中得到广泛应用并获得了良好收益，但是，一些不足之处也逐渐显现，其发展主要面临如下挑战：

1）预训练代价大。预训练模型的规模呈现逐步 扩大的趋势，虽然其能力越来越强，但模型的参数也越来越多，因此，对计算设备的算力提出了更高的要求。预训练语言模型如何使用较小的代价来高质量地完成 NLP 任务是其当前面临的问题之一。

2）深层结构化语义分析存在明显的性能不足。当前大多数引入了语义知识的预训练语言模型仍难以抵御文本对抗攻击，当面向非规范和专用领域文本时，分析精度大幅降低，说明模型未能完全学习语言的真正寓意。仅在通用语料库上进行训练的模型往往经验不足，无法较好完成常识和推理任务，不符合人类使用语言的交际意图。

3）形式化知识在语言模型中存在明显构成缺失。该问题导致在问答任务中无法基于常识给出正确的回答，不足以完成细粒度的关系抽取，命名实体识别精度不高以及在知识图谱的知识系统中缺少动作、行为、状态以及事件逻辑关系的形式化描写。

4）跨模态的语言理解存在明显融通局限。跨模
态关系因缺乏深层结构化语义分析和世界知识导致推理能力较弱，造成语言理解形意相离。

与此同时，知识感知的预训练语言模型已在现阶段部分知识驱动的 NLP 下游任务中崭露头角。预训练语言模型未来可能的研究方向如下：

1）降低预训练语言模型的训练成本。当前已有
研究人员致力于在损失较小精度的前提下探索更小、更快的预训练语言模型，当前提供的思路包含对模型采用压缩和知识蒸馏等方法，如百度公司发布的 ERNIE-Tiny，将 ERNIE 2.0 Base 模型进行压缩，取得了 4.3 倍的预测提速。此外，研究新的高效的模型架构也将对降低训练消耗起到积极作用。

2）从海量的非结构化文本而非 infobox 中搜集系统的非结构化知识，构建新型多元知识图谱增强的预训练语言模型。在实体的静态事实描述较为充分的基础上，进一步完善事件的动态描述以及事件之间逻辑关系的构建。

3）未来可向大数据与富知识共同驱动的方向进行研究，并通过与图像等跨模态信息进行交互，提升以自然语言为核心的中文语义理解能力，在实践中探索经验主义与理性主义相结合的自然语言处理新范式。

参 考 文 献

[1] Zhang, Zhengyan, et al. ERNIE: Enhanced Language Representation with Informative Entities[J]. ACL ,2019,1441-1451.

[2] Liu, Weijie, et al. K-Bert: Enabling Language Representation With Knowledge Graph[J].THIRTY-FOURTH AAAI CONFERENCE ON ARTIFICIAL INTELLIGENCE ,2020,2901-2908.

[3] Sun, Tianxiang, et al.CoLAKE: Contextualized Language and Knowledge Embedding[J]. COLING,2020,3660-3670.