LEBERT

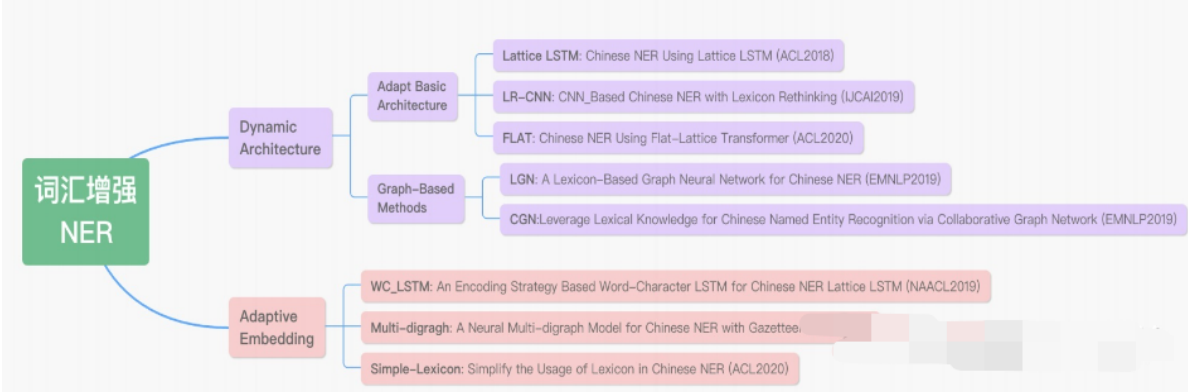
> 基于词汇增强的中文 NER 模型
>
> Lexicon Enhanced Chinese Sequence Labelling Using BERT Adapter

1.1 词增强

中文 NER 的词汇增强主要分为两条路线：

（1）**Dynamic Architecture**：通过动态结构，在模型中注入词汇信息。

（2）**Adaptive Embedding:** 将词汇信息融合到 **Embeding** 中。

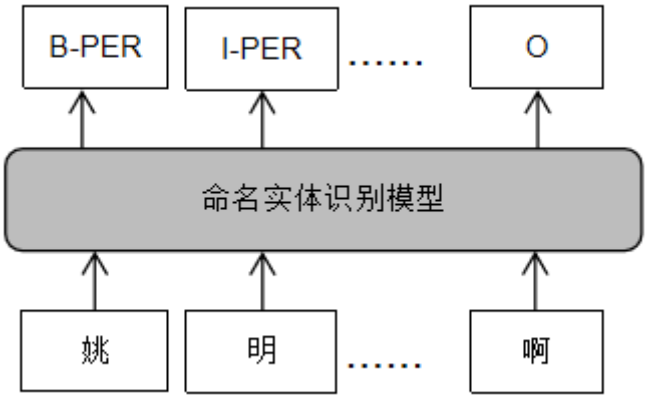


其具体实现方法总结为

	模型名称	简介
Dynamic Architecture	Lattice LSTM	通过Lattice结构将词汇信息注入模型
	LR-CNN	多尺度CNN提取特征，回溯机制解决词汇冲突
	CGN	协作图网络，融合多个图结构
	LGN	全局与局部聚合更新的图网络
	FLAT	利用Transformer进行encode，相对位置编码融合词汇信息
Adaptive Embedding	WC-LSTM	四种策略进行静态的固定编码
	Multi-digraph	使用多图结构建模字符和词典的联系
	Simple-Lexicon	使用Soft-lexicon，将词汇信息融入

0.1 任务概述

命名实体识别（简称 NER）是 NLP 中的经典任务，即给定一个输入文本，让模型识别出文本中的实体信息。

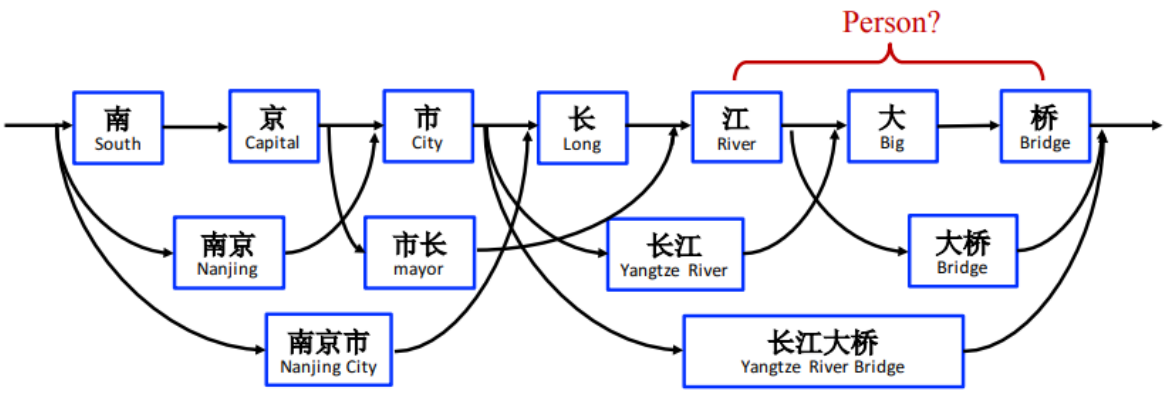


在中文 NER 任务中，可以分为 Character-based (字符粒度) 和 Word-based (词语粒度) 两种方法。

1. Word-based (词语粒度) ：基于词语粒度的中文 NER 任务，首先会对输入文本进行分词，将每个词语视为一个 token，模型对每个词语进行打标签，从而识别出实体。该方法存在一个天然的缺陷， 分词的效果严重影响模型的效果，如果分词的边界出现了问题，则模型无论如何也无法识别出正确的实体。
2. Character-based (字符粒度) ：基于字符的中文 NER 模型不需要对输入进行分词，输入中的每个汉字表示一个 token，不存在分词边界的问题(正确地将连续的自然语言文本分割成有意义的词语序列的问题)。然而该类方法无法充分 利用词汇信息，但词汇信息对于实体识别却又至关重要，它能够引入汉字之间的关联性，辅助我们进行实体边界判断。

> 对于英文句子“Time flies like an arrow”，可能有多种分词方式，如“Time”, “flies”, “like”, “an”, “arrow”、“Time”, “flies”, “like”, “a”, “narrow”等等，每一种分词方式都可能导致不同的语义解释。因此，分词边界问题是自然语言处理中一个非常重要且具有挑战性的问题。

从下图的例子【南京市长江大桥】中，我们可以看到，若只以汉字为粒度进行 NER 任务，必然会缺失大量的先验的词汇知识。如果我们能够将这些词汇信息融入到模型中，想必能够提升模型的性能.

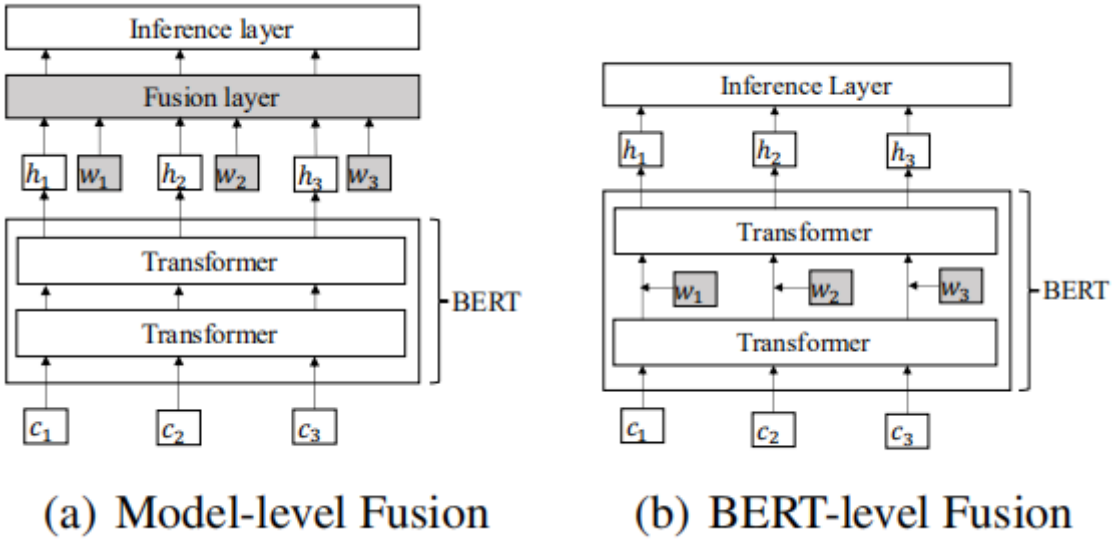


基于上述原因，目前在中文 NER 任务中，**Character-based (字符粒度) 的 NER** 做法更为普遍。但由于该类方法缺少重要的词汇信息，所以目前很多方法都致力于将词汇信息引入模型中。

0.2 模型概述

目前为止，有许多工作尝试将 BERT 与词汇特征进行结合，来进行中文的 NER 任务。当前较为普遍做法如下图（a）所示，首先使用 BERT 对字符序列进行建模，捕获字符之间的依赖关系，然后将 BERT 输出的字符特征与词汇特征进行融合，最后输入到神经网络标注模型.

该方法在一定程度上能够将词汇特征引入序列标注模型中，但由于仅在 **BERT** 末端的浅层神经网络中引入词汇特征，没有充分地利用上 **BERT** 模型的序列建模能力。因此，作者提出了在 BERT 的底层注入词汇特征的方法，模型整体示意图如下图（b）所示



(a) Model-level Fusion (b) BERT-level Fusion

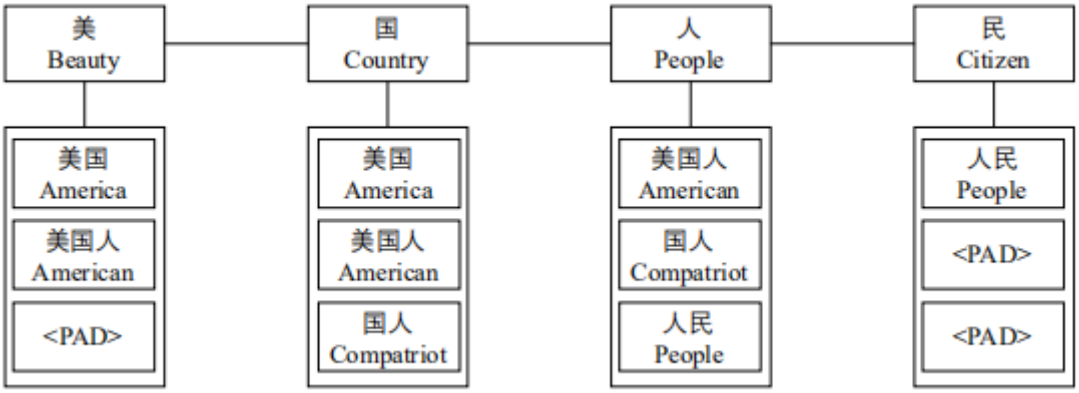
就整体结构而言，模型的主要创新点如下：

1. 引入了字符 - 词语对（ Char-Words Pair ）作为模型的输入特征。
2. 基于模型适配器的思想，设计了一种 Lexicon Adapter 的结构，将词语特征与字符特征进行融合。

0.3 Char-Words Pair Sequence

为了在模型中引入词语特征，作者设计了一种字符 - 词语对（Char-Words Pair）的结构，对于输入文本中的每个字符，找出它在输入文本中匹配到的所有词语。

对于下图中的输入文本【美国人民】，字符【美】匹配到的词语为【美国、美国人】，字符【国】匹配到的词语为【美国、美国人、国人】，以此类推。其中，作者会提前构建一棵字典树（**Trie** 树），用来匹配输入序列中的词语，然后得到每个字符对应的词语序列



给定长度为 n 的输入序列 $s_c = \{c_1, c_2, \ldots, c_n\}$其中$c_i$表示输入序列中的第 i 个字符。使用字典树进行词语匹配，得到每个字符对应的词语列表$ws = \{ws_1, ws_2, \ldots, ws_n\}$其中$ws_i$表示第 i 个字符匹配到的词语列表。最终得到字符 - 词语对 $s_{cw} = \{(c_1, sw_1), (c_2, sw_2), \ldots, (c_n, sw_n)\}$ 作为模型的输入。

0.4 Lexicon Adapter

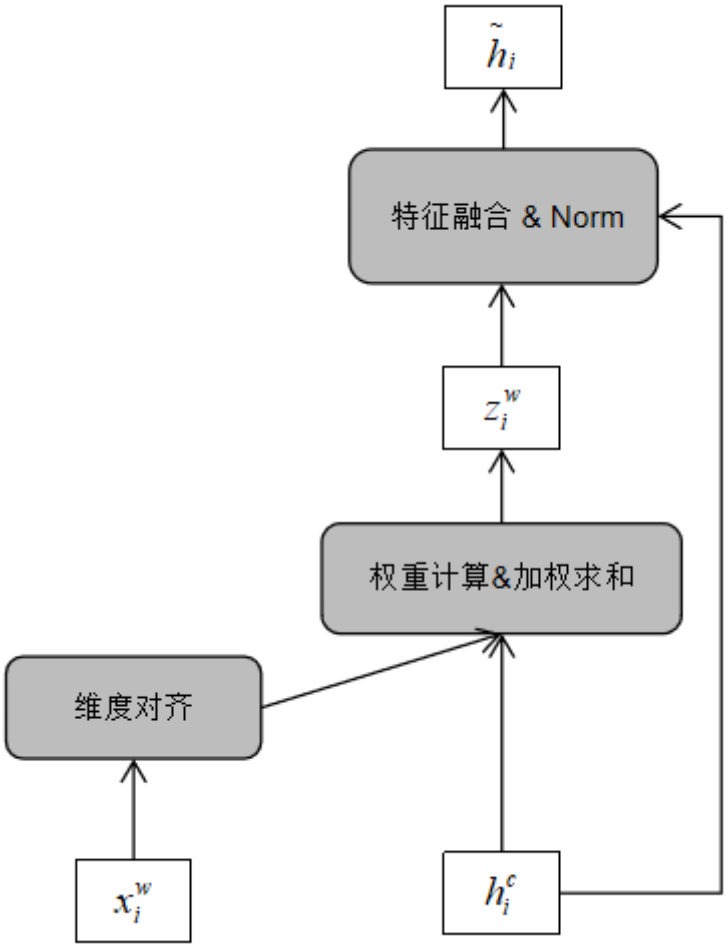
对于每个位置，都包含了两种特征：字符特征与词语特征。为了将这两种特征进行融合，作者设计了一种 Lexicon Adapter 的结构，在第 i 个位置，Lexicon Adapter 的输入表示为 (h_i^c, x_i^w)，这是个字符特征 - 词语特征对。

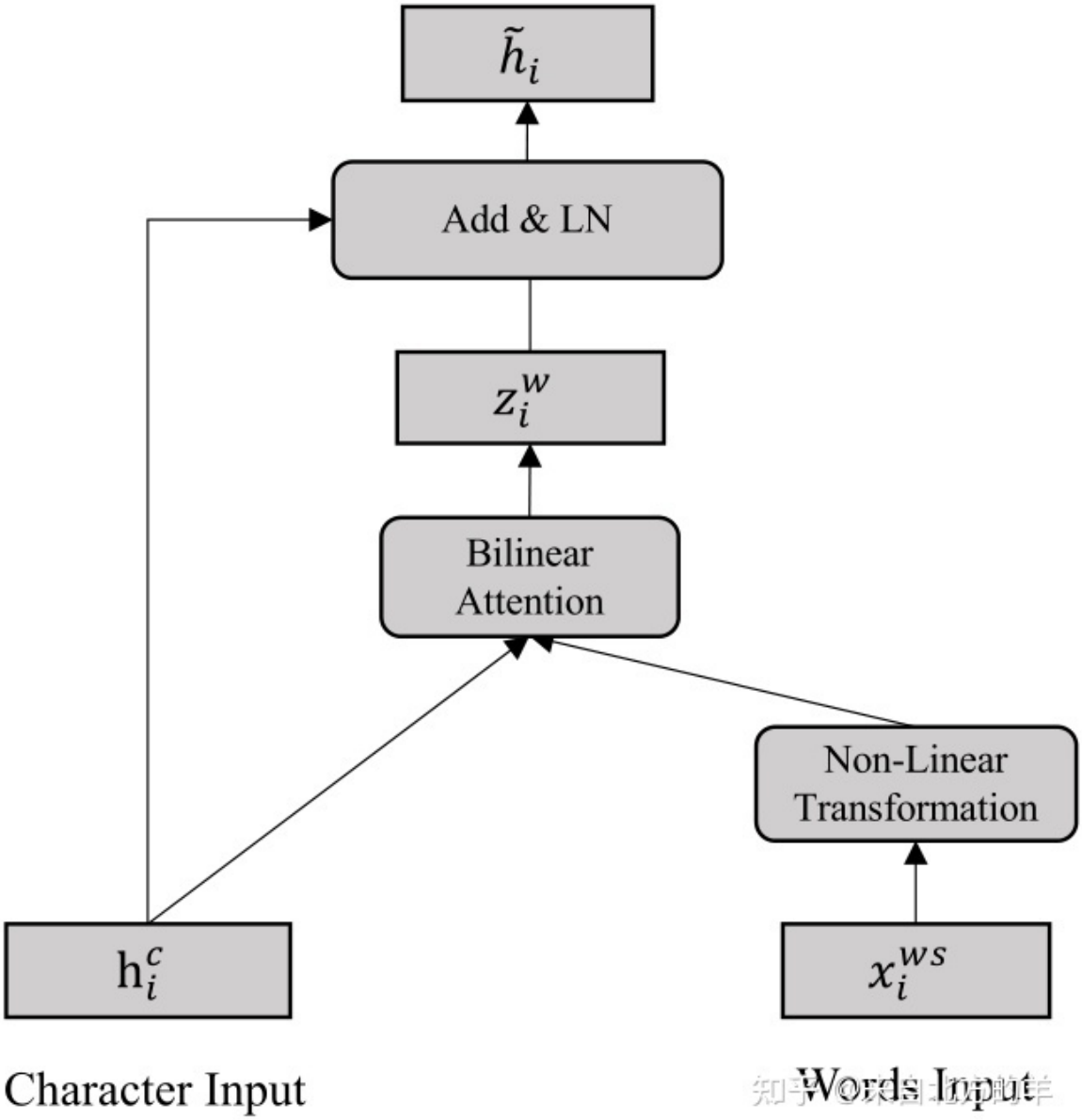
1. 字符特征 ：其中 h_i^c 可以视为第 i 个字符的特征，表示由 BERT 的某一个 transformer layer 所输出的第 i 个字符的向量表征。具体由哪一个 transformer layer 输出可以自行设定，该问题会在论文的实验部分进行讨论。

2. 词语特征 ： $x_i^w = \{x_{i1}^w, x_{i2}^w, \ldots, x_{im}^w\}$ 是一个词向量列表， x_{ij}^w 表示第 i 个位置的字符匹配到的第 j 个词语的词向量。

Lexicon Adapter 的大致可以分为以下几个步骤：

1. 维度对齐：将词向量与字符向量进行维度对齐。
2. 权重计算：对于每个字符，计算它所匹配到的每个词向量的权重。
3. 加权求和：对于每个字符，将词向量进行加权求和，得到该字符的加权词语向量。
4. 特征融合：字符向量与加权词语向量相加，得到 Lexicon Adapter 的输出。


$\tilde{h}_i$
特征融合 & Norm
z_i^w
权重计算&加权求和
维度对齐
x_i^w
h_i^c



0.4.1 维度对齐：

由于字符向量h_i^c 与词向量 x_{ij}^w的维度不一致，所以首先将词向量进行非线性映射，将其与字符向量进行维度对齐。

$$v_{ij}^w = W_2\left(\tanh\left(W_1 x_{ij}^w + b_1\right)\right) + b_2$$

其中$W_1 \in \Re^{d_c * d_w}$，$W_2 \in \Re^{d_c * d_c}$，d_c为字符特征的维度,d_w为词向量的维度。

0.4.2 权重计算：

对于每个字符，其词语列表中的每个词语的重要程度是不一样的，所以需要对每个词向量计算权重。令

$V_i = \{v_{i1}^w, v_{i2}^w, \ldots, v_{in}^w\} \in \Re^{m * d_c}$，则每个词语的权重计算方式如下

$$a_i = \text{softmax}\left(h_i^c W_{attn} V_i^T\right)$$

其中$a_i = \{a_{i1}, a_{i2}, \ldots, a_{im}\}$,$a_{ij}$表示第 i 个字符的第 j 个词向量的权重，$W_{attn}$表示双线性注意力权重矩阵。

0.4.3 加权求和：

对于每个字符，根据其匹配到的每个词语的权重，对词向量进行加权求和，得到该字符的加权词语向量

z_i^w。

$$z_i^w = \sum_{j=1}^{m} a_{ij} v_{ij}^w$$

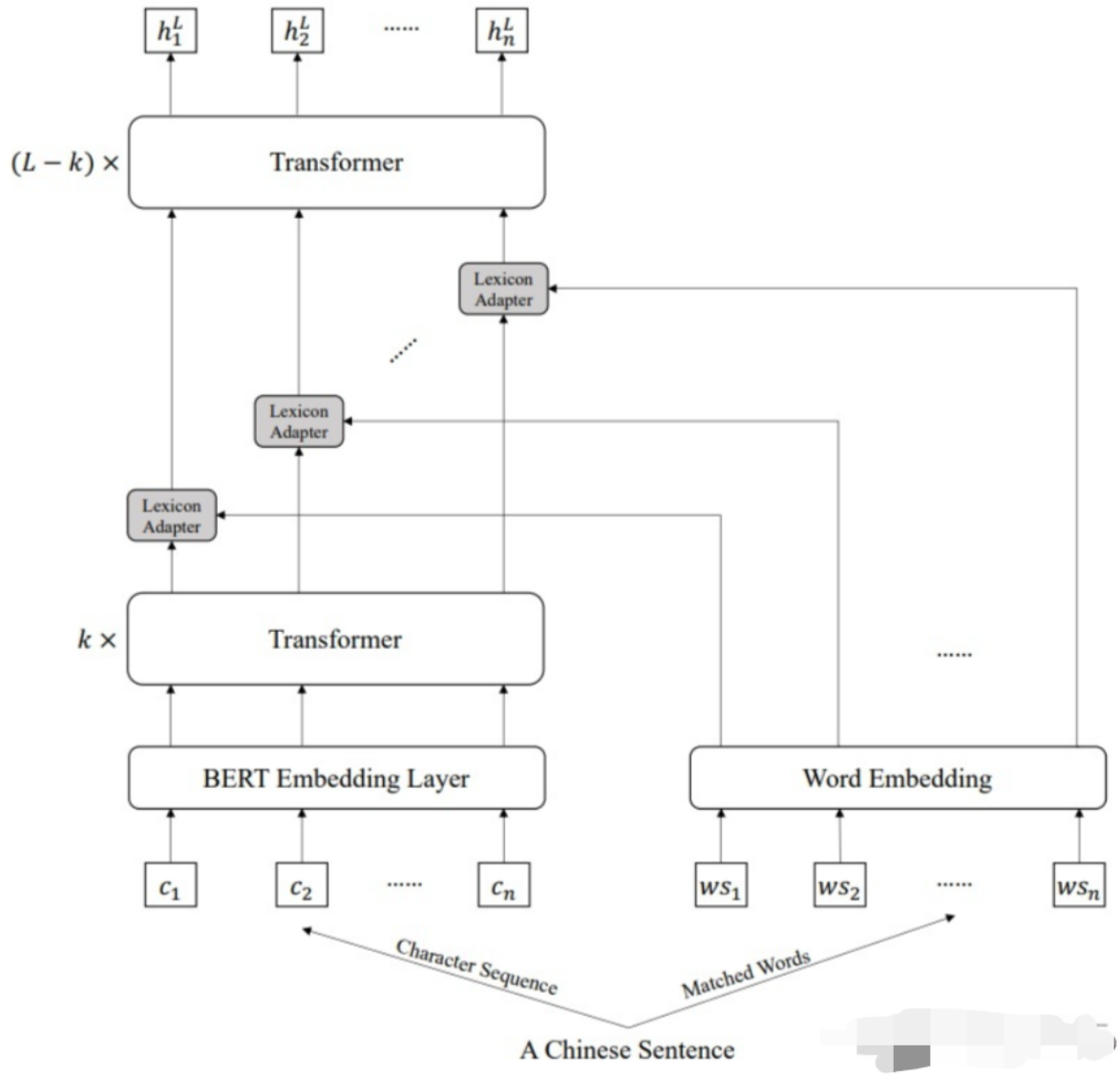
0.4.3.1 特征融合：

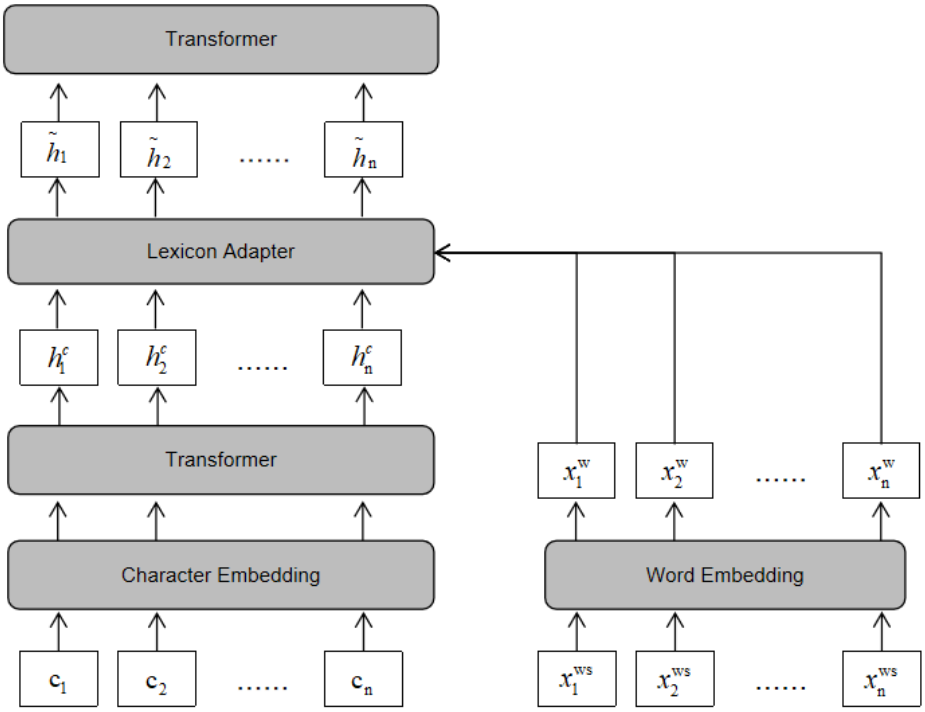
将字符向量与加权词语向量相加，得到特征融合向量$\tilde{h}_i$，最后将$\tilde{h}_i$进行 dropout、layer norm 和残差连接等操作，得到 Lexicon Adapter 的最终输出。

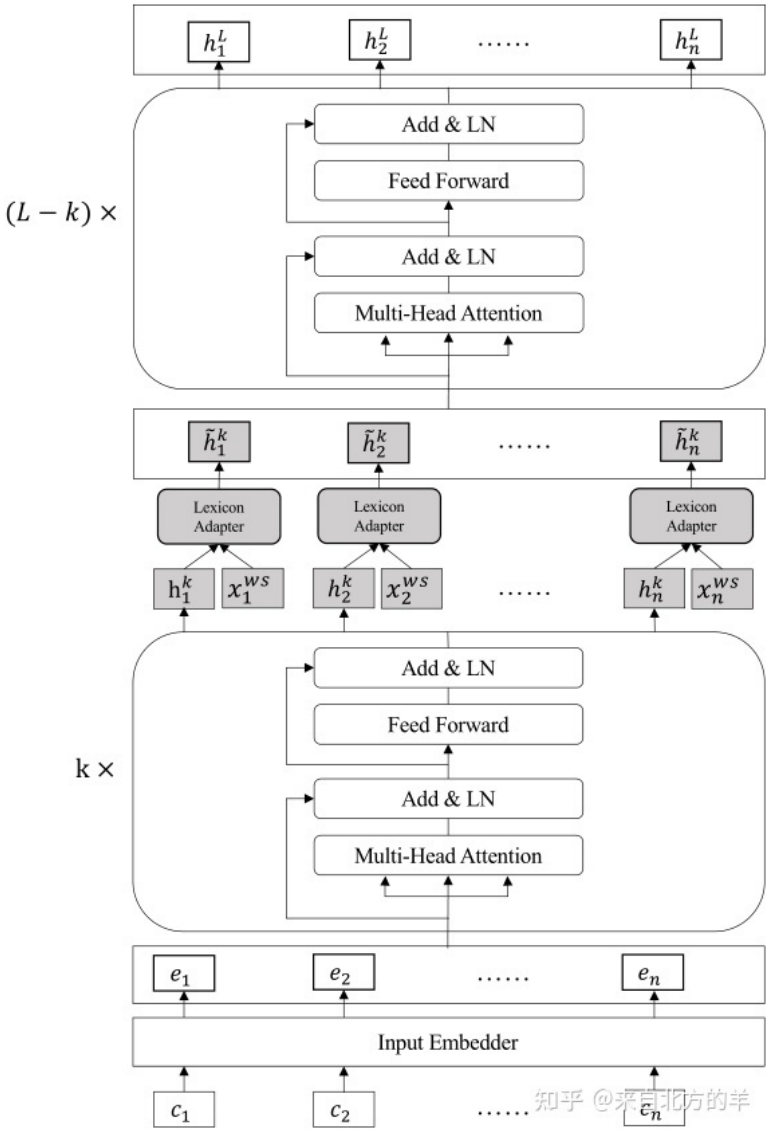
$$\tilde{h}_i = h_i^c + z_i^w$$

0.5 Lexicon Enhanced BERT

如下图所示，将 Lexicon Adapter 与 BERT 相结合，得到最终的模型结构。其中 Lexicon Adapter 可以插到任意一个 Tranformer Layer 的后面进行词语特征融合。经过作者的消融实验，发现 **Lexicon Adapter** 插到第一个 **Tranformer Layer** 后面的效果最好。




Transformer
Lexicon Adapter
Transformer
Character Embedding
Word Embedding


$(L-k)\times$
Add & LN
Feed Forward
Add & LN
Multi-Head Attention
Lexicon Adapter
Lexicon Adapter
Lexicon Adapter
k ×
Add & LN
Feed Forward
Add & LN
Multi-Head Attention
Input Embedder

LEBERT 结构图上图所示，其可以看做是 Lexicon Adapter 和 BERT 的组合，其中 Lexicon Adapter 应用到了 BERT 当中的某一个 Transformer 层。

步骤：

- 对于给定的中文$s_c = \{c_1, c_2, \ldots, c_n\}$将其构建成 character-words pair sequence 形式 $s_{cw} = \{(c_1, ws_1), (c_2, ws_2), \ldots, (c_n, ws_n)\}$
- $\{c_1, c_2, \ldots, c_n\}$ 输入到Bert encoder.每个 Transformer encoder 表示为如下形式：

$$
\begin{aligned}
G &= \text{ Layernormalization } \left(H^{l-1} + \text{ Multiheadattention } \left(H^{l-1}\right)\right) \\
H^l &= \text{ Layernormalization } (G + FFN(G))
\end{aligned}
$$

之后，通过 Lexicon Adapter 把词汇信息注入到第 k 层和第 k+1 层 Transformer 层之间。

- k 层 Transformer 层的输出为$H^k = \left\{h_1^k, h_2^k, \ldots, h_n^k\right\}$。将其中的每一个 Char-Words Pair

  $\left(h_i^k, x_i^{ws}\right)$利用 Lexicon Adapter 进行转换得到：$\tilde{h}x_i^k = LA\left(h_i^k, x_i^{ws}\right)$然后将注入词汇信息的特征向量输出入到余下的 L-k 个 Transformer 层中。

  作者在之前的基础上加了一个线性变换层和 CRF 层来进行标签解码, 从而得到每个字符的 label。

0.6 实验细节

本项目的实验设置如下：

1. 所有模型均使用 bert-base-chinese 的预训练权重。
2. 本项目将词向量信息在 BERT 的第一层之后进行融合，并且每个汉字，最多融合 3 个词向量。
3. 在训练的时候，batch size 均为 32，BERT 的原始参数的学习率设置为 1e-5，LEBERT 和 CRF 所引入的参数的学习率设置为 1e-4。
4. 对于 Ontonote、Msra 数据集训练 10 个 epoch， 对于 Resume 和 Weibo 数据集训练 30 个 epoch。
5. 原论文的词向量使用的是包含两千万词语的 tencent-ailab-embedding-zh-d200-v0.2.0。本项目使用的词向量为 tencent-ailab-embedding-zh-d200-v0.2.0-s，其中包含两百万预训练的词向量，维度为 200。