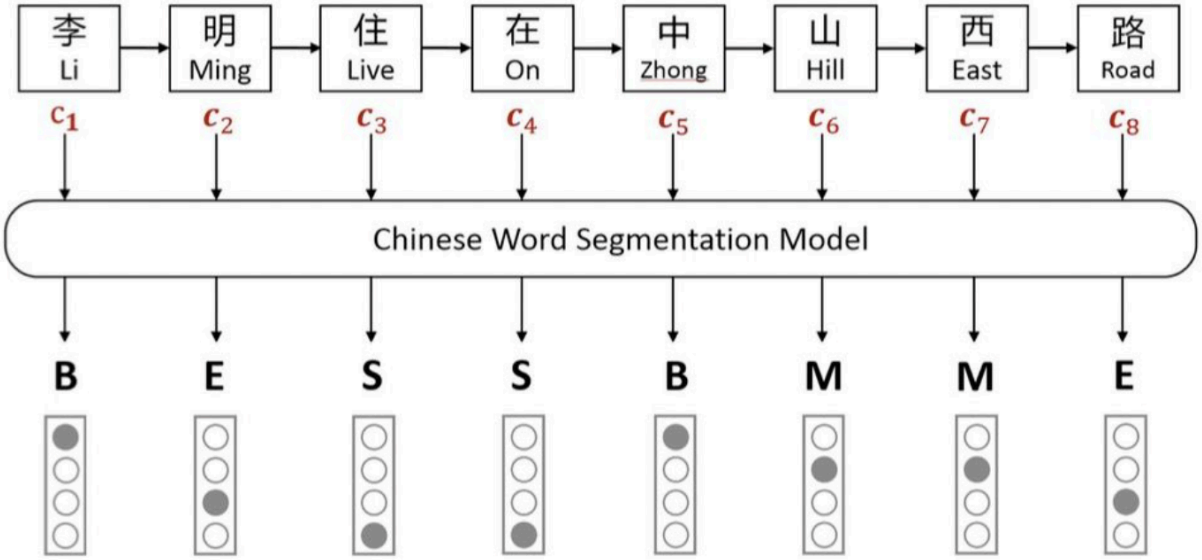
Softword

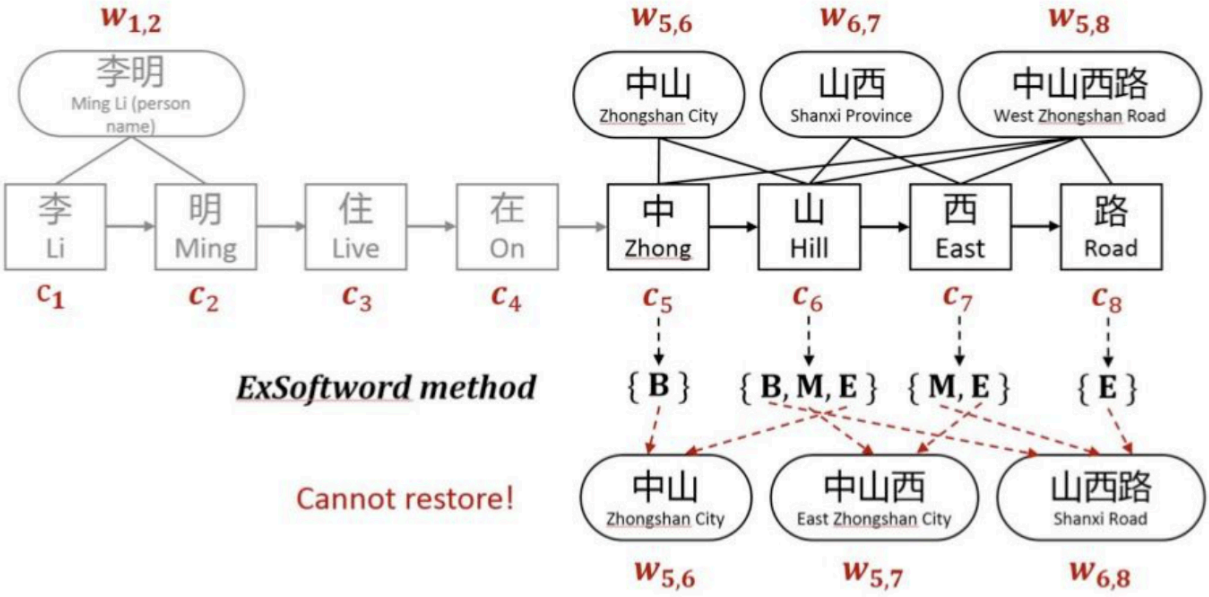


将文本通过中文分词模型，根据分词结果得到每个字符的 B/M/E/S 标签。在模型输入侧，把分词的 label encoding 进行向量表达，采用相加或者拼接的方式，加入到已有的 token embedding 上。

这里在实现的时候，分词模型使用 jieba；B/M/E/S 标签对应的 embedding 使用随机初始化的 embedding 向量。

这种方式引入了词边界信息，对于模型的效果的提升有一定的帮助，但是没有引入词语本身的语义信息，提升效果有限。同时，存在分词造成的误差传播问题。

ExtendSoftword



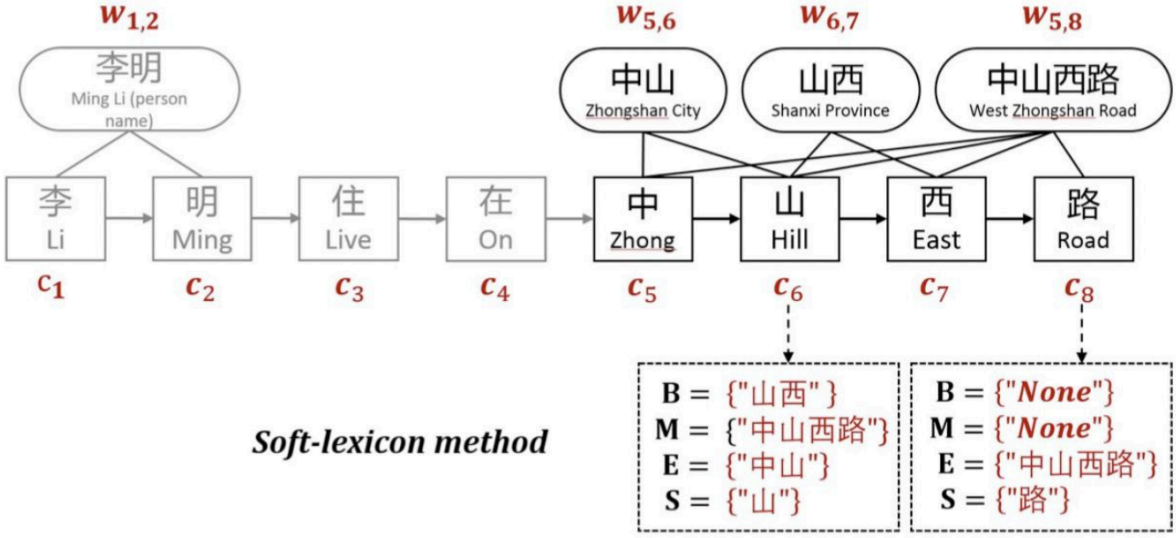
ExtendSoftword 则将所有可能匹配到的词汇结果对字符进行编码表示。如上图中，对于字符「山」来说，其可匹配到的词汇有中山、中山西、山西路，「山」隶属于 $\{B, M, E\}$ 。对于字符对应的词汇信息按照 B/M/E/S/O 编码构建 5 维二元向量，如 「山」 表示为 $[1, 1, 1, 0, 0]$。

在实现的过程中，每个字符会遍历词表，拿到所有包含这个字且满足前后文本的所有分词 tag。整个遍历的复杂度是 $O(NK)$，其中，N 是句子长度，K 是最长词搜索长度，实现中取 5。词表的来源可以是对预训练词向量进行清洗，如去除单个字符和过长的词。

在模型输入侧，将 5 维二元向量与对应的 embedding 矩阵相乘，得到对应的 embedding 表示，与字符本身的 embedding 拼接/相加。

缺点依旧是没有引入词汇本身的语义信息。

SoftLexicon



SoftLexicon 是在 ExtendSoftword 的基础上把只使用 B/M/E/S 的分词 tag，变成直接使用所有匹配上的单词。每个字符都得到该字符作为 B/M/E/S 所能匹配上的所有单词，这样引入词边界信息的同时也引入词汇本身的信息。

在实现过程中，由于每个字符的 B/M/E/S 对应的词汇数量不一致，需要 pad 或 trunc 到统一的长度。当前字符引入词汇信息后的特征表示为：

$$e^s(B, M, E, S) = [v^s(B) \oplus v^s(M) \oplus v^s(E) \oplus v^s(S)]$$

$$\mathbf{x}^c \leftarrow [\mathbf{x}^c; \mathbf{e}^s(B, M, E, S)]$$

上述公式将 B/M/E/S 对应的词汇编码 v^s 与字符编码 x^c 进行拼接。v^s 表示当前标签 B/M/E/S 下对应的词汇集合编码，其计算方式为：

$$v^s(S) = \frac{1}{Z} \sum_{w \in S} z(w) \mathbf{e}^w(w)$$

S 为词汇集合，$z(w)$ 代表词频；词频可以通过统计数据集得到。

SoftLexicon 相比上述两个方法，不仅引入和词汇的语义信息，而且考虑了词汇的重要性（词频）。这个方法的一个好处与模型无关，可以适配于其他序列标注框架。在 BiLSTM 模型上效果提升明显，在 Bert 模型上也有一定的提升。

参考资料

1. https://zhuanlan.zhihu.com/p/142615620
2. https://www.cnblogs.com/gogoSandy/p/14965711.html