



北京大學

# 本科生毕业论文

PolicyBERT: 基于全词掩码的中文政策文本预训练语言模型

PolicyBERT: A Chinese Policy Text Pre-trained Language Model Based on Whole Word Masking

姓名: 浦皓天

学号: 2100016620

院系: 信息管理系

专业: 大数据管理与应用

导师姓名: 孟凡

二〇二五年五月

## 摘要

关键词:

## Abstract

**Keywords:**

# 目录

<b>1</b>	<b>引言</b>	<b>5</b>
<b>2</b>	<b>相关研究</b>	<b>7</b>
<b>3</b>	<b>PolicyBERT</b>	<b>8</b>
3.1	研究方法 . . . . .	8
3.1.1	模型输入 . . . . .	8
3.1.2	嵌入、编码与融合 . . . . .	10
3.1.3	预训练头 . . . . .	11
3.1.4	下游任务头 . . . . .	13
3.2	实验准备 . . . . .	13
3.2.1	数据集 . . . . .	13
3.2.2	数据预处理 . . . . .	14
3.2.3	微调数据集构建 . . . . .	15
3.2.4	骨干模型 . . . . .	16
3.2.5	训练配置 . . . . .	17
3.3	实验结果 . . . . .	17
3.3.1	PolicySM-1 . . . . .	17
3.3.2	PolicySM-2 . . . . .	19
<b>4</b>	<b>未名问政</b>	<b>21</b>
4.1	系统架构 . . . . .	21
4.2	系统演示 . . . . .	22
<b>5</b>	<b>结论</b>	<b>23</b>

# 1 引言

自然语言处理（Natural Language Processing, NLP）是人工智能领域的重要分支，旨在使计算机能够理解、分析和生成自然语言。随着互联网的快速发展，海量文本数据的产生为 NLP 研究提供了丰富的资源，同时也带来了新的挑战。尤其是在中文文本处理方面，由于汉字的独特性和复杂性，传统的基于词典的方法难以满足实际需求。因此，如何有效地表示和理解中文文本成为了一个亟待解决的问题。

在自然语言处理的早期，文本表示主要依赖于基于词典的特征工程方法，如 TF-IDF 通过词频和逆文档频率量化词的重要性 [1]，主题模型（如 LDA）则挖掘文本的潜在主题分布，将文档映射到低维主题向量空间，提升了语义理解能力 [2]。随着深度学习的兴起，分布式词向量技术（如 Word2Vec [3] 和 GloVe [4]）通过大规模无标注语料训练，将词映射为低维实数向量，捕捉词间相似性，但其“静态”表示难以区分多义词，对上下文变化不敏感。为此，ELMo 引入了基于双向语言模型的上下文相关表示，使同一词在不同上下文中具有不同向量，显著提升了下游任务性能 [5]。

基于 Transformer 架构的预训练语言模型（PLM）带来了革命性变革。BERT 首创了“掩码语言模型”和“下游任务微调”范式，通过海量语料预训练并在具体任务上微调，大幅提升性能，奠定了通用 NLP 研究的基础 [6]。其后，RoBERTa 通过更大规模语料和更长训练时间刷新了多项任务表现 [7]，ERNIE 则融入知识图谱与实体信息，在预训练中引入外部知识，进一步增强了语义理解能力 [8]。

尽管上述预训练模型在英文 NLP 任务中表现优异，直接迁移到中文时却面临语言结构层面的挑战。中文语言天然缺乏空格作为词语边界标记，模型在对文本进行编码前通常需先进行分词处理。但中文的词边界存在模糊性和上下文依赖性，使得不同分词方式可能产生完全不同的语义理解。例如，“北京大学”可被切分为“北京/大学”或整体作为专有名词“北京大学”，两者所表达的语义截然不同。这种分词歧义会导致 token 级别的预训练模型无法准确建模真实语义。

现有中文 BERT 模型通常采用字符级（字粒度）编码策略，即将每个汉字视为独立 token，这虽然规避了分词带来的歧义问题，但也牺牲了对词语整体语义的把握。例如，将“人工智能”拆解为“人”“工”“智”“能”，使模型需要通过更长的上下文依赖才能捕捉到其作为专有名词的统一语义。这种处理方式尤其不利于处理术语密集、长词多、句式复杂的文本类型。

在 NLP 的具体应用中，领域文本（Domain-specific Text）处理逐渐成为研究热点。尤其是法律、金融、医疗、政策等垂直领域，其文本具有术语密集、专业性强、结构复杂、长距离依赖、命名实体多、更新频繁等特点，对文本编码提出了更高要求：

模型不仅要能理解个体 token 的语义, 还要具备对领域词语、命名实体的整体感知能力。这使得纯粹依赖字粒度编码的 BERT 类模型难以充分捕捉政策文本中的关键信息。若模型不能准确识别和建模“国家政策”“财政转移支付”“扶贫开发”等多字短语或专有名词, 可能导致信息丢失、下游任务性能下降

ZEN (a BERT-based Chinese (Z) text encoder Enhanced by N-gram representations) 提出了引入 n-gram 表示的中文词信息 [9]。该文使用了 Shizhe Diao 等人提出的基于计算相邻字符 PMI (Pointwise Mutual Information) 的分词方法 [10]。将所有连续的 n-gram 片段与字符表示并行编码, 并在模型的每一个交叉注意力组件的末尾进行相加融合, 以补充多字组合信息。

本文将在 ZEN 的基础上, 提出一种新的中文政策文本预训练语言模型 PolicyBERT。该模型在 ZEN 的基础上引入了 HanLP 分词、门控与注意力机制, 以增强对中文政策文本的建模能力。

随着自然语言处理技术的快速发展, 基于检索增强生成 (Retrieval-Augmented Generation, RAG) 的系统架构在政策文本处理领域展现出巨大潜力。RAG 是一种将大型语言模型与外部知识库相结合的方法, 研究表明, RAG 在多个评估指标上优于传统的领域特定微调方法, 尤其在知识检索任务中表现出更高的准确性和可靠性 [11]。政策文本通常具有术语密集、结构复杂、领域专有性强等特点, 这对模型的理解和生成能力提出了更高要求。传统的预训练语言模型虽然在通用任务上表现优异, 但在特定领域的任务中往往难以充分捕捉领域知识。因此, 在子领域上微调过的预训练模型成为解决这一问题的关键, 能够更好地适应特定领域的语义需求, 提升 RAG 系统的性能。

Ollama 是一个高效的模型部署工具, 支持本地化部署大规模语言模型, 能够显著降低对外部依赖的需求。本文中, 我们使用 Ollama 实现本地部署 *Qwen2.5 : 7b*, 这是一款由阿里巴巴达摩院推出的强大中文语言模型, 具备卓越的生成能力和对中文语义的深度理解 [12]。此外, BGE (BAAI General Embedding) 系列模型在 RAG 任务中表现出色, 尤其是 *bge-base-zh-v1.5*, 其在中文文本嵌入生成方面具有较高的语义一致性和检索精度, 为构建高效的检索模块提供了坚实基础 [13]。

在架构设计上, langchain-chatchat 是一种灵活且高效的框架, 专为复杂的 RAG 系统设计, 能够无缝集成检索和生成模块, 支持多阶段任务处理 [14]。langchain-chatchat 的优势在于其模块化设计和对大规模模型的兼容性, 使得系统能够在保持高性能的同时, 具备良好的扩展性和易用性。

本文将结合 Ollama 部署的 *Qwen2.5 : 7b* 和微调后的 *bge-base-zh-v1.5*, 构建一个高效的 RAG 系统——“未名问政”。该系统将利用本地部署的 *Qwen2.5 : 7b* 进行文本生成, 并通过使用 ZEN 架构微调后的 *bge-base-zh-v1.5* 实现高效的政策文本检索与生成任务。通过这一系统, 本文验证了在特定领域微调模型和优化架构设计对提升 RAG 系统性能的有效性。

本文的主要贡献包括以下几个方面: 提出 PolicyBERT 架构, 引入 HanLP 分词、门控与注意力机制, 以增强对中文政策文本的建模能力; 将 *bge-base-zh-v1.5* 的主体 BERT 部分参数与预训练参数拼接, 使用 PolicyBERT 架构进行微调, 以适应政策文本的特定需求; 使用 Ollama 部署 *Qwen2.5 : 7b*, 并结合 langchain-chatchat 架构构建 RAG 系统, 利用微调后的 *bge-base-zh-v1.5*, 实现高效的政策文本检索与生成任务。通过这一系统, 本文验证了多层融合词级信息对文本理解效果的提升, 以及在特定领域微调模型和优化架构设计对 RAG 系统性能的提升。本文 PolicyBERT 的训练和未名问政 RAG 系统的代码公开在<https://github.com/PuHT4213/PolicyBert>和<https://github.com/PuHT4213/WeiMingPolicyRAG>。

## 2 相关研究

ZEN (Pre-training Chinese Text Encoder Enhanced by N-gram Representations) 提出了一种专为中文设计的预训练语言模型, 旨在解决中文文本处理中的分词不确定性问题 [9]。相较于将每个汉字作为独立 token 的传统 BERT 模型, ZEN 引入 n-gram 表示, 通过显式建模多粒度的词片段信息, 增强了对词级语义的建模能力。在架构上, ZEN 在字符级 BERT 的基础上添加了 n-gram 编码器, 通过融合字符与 n-gram 表示, 有效提升了模型对中文语义结构的理解。实验证明, ZEN 在中文分词、命名实体识别、文本分类等多个任务上优于传统方法, 表明其多粒度建模策略在中文 NLP 中具有重要价值。

PTWA (Pre-Training with Word Attention) 提出了一个针对中文命名实体识别 (NER) 任务的预训练方法, 旨在更有效地融合词级信息以提升模型性能 [15]。该方法通过引入词注意力机制, 增强了字符表示与词级语义之间的交互, 从而更准确地识别实体边界和类别。在实验中, PTWA 在多个中文 NER 基准数据集上表现出色, 显著优于传统的字符级预训练模型, 验证了其在中文 NER 任务中的有效性和先进性。

Wenbiao Li 等人 (2022) 在其研究《Exploiting Word Semantics to Enrich Character Representations of Chinese Pre-trained Models》中, 针对中文预训练语言模型

普遍采用字符级建模而忽略词级语义的问题，提出了一种名为“HRMF”（Hidden Representation Mix and Fusion）的新方法 [16]。该方法通过将词嵌入按相似度权重投射到其内部字符嵌入中，增强了字符表示的语义信息。随后，利用字符间的混合机制强化词边界信息，并引入词-字符对齐注意力机制，突出重要字符，抑制不重要字符的影响。此外，为减轻分词错误传播的问题，作者设计了融合多种分词器结果的集成策略。实验结果显示，该方法在情感分类、句子对匹配、自然语言推理和机器阅读理解等任务中，显著优于基础的中文预训练模型如 BERT、BERT-wwm 和 ERNIE，验证了其在丰富字符表示和提升模型性能方面的有效性。

Qiang He 等人 (2023) 在其论文《Prompt-Based Word-Level Information Injection BERT for Chinese Named Entity Recognition》中，提出了一种名为 PWII-BERT 的模型，旨在提升中文命名实体识别（NER）的性能 [17]。该模型通过引入提示（prompt）机制，将词级信息注入预训练语言模型中，从而更有效地捕捉字符与词之间的关联。具体而言，PWII-BERT 设计了一个词级信息注入适配器（WIIA），利用双仿射机制融合字符和词的特征，并在 Transformer 层中引入类别提示，指导模型关注特定类别的实体信息。实验结果显示，PWII-BERT 在四个中文 NER 基准数据集上均优于现有的主流模型，验证了融合类别信息和词汇特征对于提升中文 NER 性能的有效性。

### 3 PolicyBERT

#### 3.1 研究方法

本文提出的 PolicyBERT 的模型架构如图所示。

##### 3.1.1 模型输入

令输入的所有政策文件定义为

$$Doc = \{d_1, d_2, \dots, d_{N^{doc}}\} \quad (1)$$

其中  $d_i$  为第  $i$  个政策文件， $N^{doc}$  为政策文件的总数。我们将每个政策文件  $d_i$  按照标点、换行符等标识分为  $N^{d_i}$  个句子。

$$d_i = \{s_1, s_2, \dots, s_{N^{d_i}}\} \quad (2)$$

假设每个句子中的词数和字数分别为  $k_w$  和  $k_c$ （暂时不考虑如 [UNK]、[CLS]、



[SEP] 等特殊符号的影响), 则每个句子  $s_j$  可以表示为:

$$\begin{aligned} s_j &= W_j = C_j \\ W_j &= \{w_1, w_2, \dots, w_{k_w}\} \\ C_j &= \{c_1, c_2, \dots, c_{k_c}\} \end{aligned} \quad (3)$$

其中  $w_i$  为第  $i$  个词,  $c_i$  为第  $i$  个字。  $k_w$  和  $k_c$  分别为句子中的词和字符的数量。字符数量  $k_c$  由句子长度决定。具体对于下游 NSP (Next Sentence Prediction) 任务, 模型输入  $s_j$  还可以表示为:

$$s_j = [CLS] + s_j^{(1)} + [SEP] + s_j^{(2)} + [SEP] \quad (4)$$

为了更好地对应词和字的关系, 我们使用一个匹配矩阵 (Matching Matrix)  $\mathcal{M}_j \in N^{k_w \times k_c}$ , 也即:

$\mathcal{M}_j$  是一个  $k_w \times k_c$  的 0-1 矩阵, 表示句子中的每一个字是否属于某个词。矩阵的每一行对应一个词  $w_q$ , 每一列对应一个字  $c_p$ 。如果字  $c_p$  属于词  $w_q$ , 则矩阵中对应位置的值为 1, 否则为 0。具体来说,  $\mathcal{M}_j$  的每一个元素的取值为:

$$m_{pq} = \begin{cases} 1, & \text{if } c_p \in w_q \\ 0, & \text{elif } c_p \notin w_q \end{cases} \quad (5)$$

实际输入模型时, 使用输入句子  $s_j$  的字符表示  $C_j$  与词表  $V$  构建匹配矩阵  $\mathcal{M}_j$  和词表示  $W_j$ 。词表  $V$  由 Hanlp 提供的分词接口预先构建, 共 2662 个词。对于  $C_j$ , 提取其中的所有存在于  $V$  的词, 记录其索引、起始位置、长度和内容。使用词在词表中的 id, 可以构建词表示  $W_j$ :

$$W_j = \{wid_1, wid_2, \dots, wid_{k_w}\} \quad (6)$$

其中  $wid_i$  为第  $i$  个词在词表  $V$  中的索引。

同时使用公式5构建上文提到的匹配矩阵  $\mathcal{M}_j$ , 最后将所有词的顺序打乱, 以避免模型学习到词表的顺序信息。每句话的最大词数和最大字符数由超参数  $k_w^{max}$  和  $k_c^{max}$  决定。对于超过的句子, 使用截断 (Truncation) 处理, 保留前  $k_w^{max}$  个词和  $k_c^{max}$  个字符。对于不足的句子, 使用零填充 (Zero Padding) 补齐。如此一来, 构成了长度为  $k_c^{max}$  词嵌入  $W_j$  和长度为  $k_c^{max}$  字嵌入  $C_j$ 。其中  $k_w^{max}$  为预先设定的单个句子中词的最大数量, 本文使用  $k_w^{max} = 40$ ; 而  $k_c^{max}$  为预先设定的单个句子中字符的最大数量, 本文使用  $k_c^{max} = 128$ 。

由此，我们构建好了模型的全部输入，包括字符表示  $C_j$ 、词表示  $W_j$  和匹配矩阵  $\mathcal{M}_j$ 。模型的输入为：

$$X_j = \{C_j, W_j, \mathcal{M}_j\} \quad (7)$$

### 3.1.2 嵌入、编码与融合

模型的输入  $X_j$  由字符表示  $C_j$ 、词表示  $W_j$  和匹配矩阵  $\mathcal{M}_j$  组成。为了将这些输入转换为模型可以处理的向量表示，我们使用了嵌入层（Embedding Layer）和编码器（Encoder）。

字符表示  $C_j$  和词表示  $W_j$  首先通过嵌入层进行转换，得到对应的嵌入向量  $C^{(0)}$  和  $W^{(0)}$ ：

$$\begin{aligned} C^{(0)} &= \text{text\_embedding}(C_j) \\ W^{(0)} &= \text{word\_embedding}(W_j) \end{aligned} \quad (8)$$

用于处理字符输入  $C_j$  和词输入  $W_j$  的模型架构相似，均为标准的 BERT 架构，使用了多层的 Transformer 编码器。每一层的输入  $C^{(l)}$  和  $W^{(l)}$ ，经过多头自注意力（Multi-head Self-Attention）处理后，再经过前馈神经网络进行非线性变换。可以表示为：

$$\begin{aligned} A^{(l)} &= \text{Attention}(C^{(l)}, \text{mask}) \\ I^{(l)} &= \text{Intermediate}(A^{(l)}) \\ O^{(l)} &= \text{Output}(I^{(l)}, A^{(l)}) \\ C_1^{(l)} &= O^{(l)} \end{aligned} \quad (9)$$

对  $W^{(l)}$  的处理方式类似，只是对于词表示的中间结果  $W_1^{(l)}$ ，会令其与匹配矩阵  $\mathcal{M}_j$  进行相乘，得到词表示中间结果  $W_2^{(l)}$ ，而对  $C_1^{(l)}$  则不进行处理。

$$\begin{aligned} W_2^{(l)} &= W_1^{(l)} \cdot \mathcal{M}_j \\ C_2^{(l)} &= C_1^{(l)} \end{aligned} \quad (10)$$

本文在此基础之上，增加了一个融合层（Fusion Layer），用于将字符和词的表示进行融合。本文使用了两种融合方式：门控机制（Gated Mechanism）和多头注意力机制（Multi-head Attention Mechanism）。

门控机制：门控机制通过一个包含全连接层（Fully Connected Layer）*Dense*、Sigmoid 激活函数和偏置项的门控单元，用字符和词的拼接表示，计算要融合哪些词

嵌入的信息。得到新的字符表示  $C^{(l+1)}$ :

$$\begin{aligned} gate^{(l)} &= \sigma^{(l)}(Dense(C_2^{(l)}, W_2^{(l)})) + b^{(l)} \\ C^{(l+1)} &= C_2^{(l)} + gate^{(l)} \cdot W_2^{(l)} \end{aligned} \quad (11)$$

其中每个线性层的偏置 (bias) 被初始化为常数  $b^{(0)} = 5.0$ , 这样通过 sigmoid 激活函数后, 初始门控值接近于 1, 这意味着初始状态下模型倾向于直接保留字符表示  $C^{(l)}$ , 与 ZEN 中直接相加  $C_2^{(l)}$  和  $W_2^{(l)}$  的原始方法处在同一出发点。

**多头注意力机制:** 多头注意力机制则是通过计算字符和词的注意力权重, 来决定如何融合两者的信息。具体来说, 对于每一层的字符表示  $C_2^{(l)}$  和词表示  $W_2^{(l)}$ , 多头注意力机制将字符表示作为查询 (Query), 词表示作为键 (Key) 和值 (Value), 输入到多头注意力模块中。通过计算注意力权重, 模型能够捕捉字符和词之间的交互关系, 从而生成融合后的表示。

具体实现如下: 首先, 使用字符表示  $C_2^{(l)}$  作为查询  $Q$ , 词表示  $W_2^{(l)}$  作为键  $K$  和值  $V$ , 输入到多头注意力模块中:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V \quad (12)$$

其中  $d_k$  是键的维度, 用于缩放点积以稳定梯度。

多头注意力机制通过多个独立的注意力头来捕捉不同的特征表示:

$$\begin{aligned} \text{MultiHead}(Q, K, V) &= \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O \\ \text{head}_i &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \quad (13)$$

其中  $W_i^Q, W_i^K, W_i^V$  是每个注意力头的参数矩阵,  $W^O$  是输出的线性变换矩阵。

在融合过程中, 模型还引入了残差连接 (Residual Connection) 和层归一化 (Layer Normalization) 以增强稳定性:

$$C^{(l+1)} = \text{LayerNorm}(C_2^{(l)} + \text{Dropout}(\text{MultiHead}(C_2^{(l)}, W_2^{(l)}, W_2^{(l)}))) \quad (14)$$

模型的最后一层输出  $C^{(L)}$ , 将被用于预训练或下游任务的微调。

### 3.1.3 预训练头

在预训练阶段, 模型的最后一层输出  $C^{(L)}$  被用于两个主要任务: 掩码语言模型 (Masked Language Model, MLM) 和下一句预测 (Next Sentence Prediction, NSP)。这两个任务分别用于增强模型的上下文理解能力和句子间关系建模能力。

对于掩码语言模型任务，模型首先对输入序列中的部分 token 进行随机掩码处理（即用特殊标记 [MASK] 替换），然后利用  $C^{(L)}$  预测这些被掩码的 token。具体来说， $C^{(L)}$  会被输入到一个语言模型预测头（Language Model Prediction Head）中，该预测头由一个全连接层和 softmax 激活函数组成，用于输出每个被掩码 token 的概率分布。预测的概率分布可以表示为：

$$P(\hat{y}_i|C^{(L)}) = \text{softmax}(\text{Linear}(C_i^{(L)})) \quad (15)$$

其中， $C_i^{(L)}$  表示第  $i$  个 token 的隐藏状态， $\hat{y}_i$  表示模型预测的第  $i$  个 token 的概率分布。

为了计算 MLM 任务的损失，采用交叉熵损失函数（CrossEntropyLoss），公式如下：

$$\mathcal{L}_{MLM} = -\frac{1}{N} \sum_{i=1}^N y_i \log(\hat{y}_i) \quad (16)$$

其中， $N$  表示被掩码的 token 数量， $y_i$  表示第  $i$  个 token 的真实标签， $\hat{y}_i$  表示模型预测的概率分布。

对于下一句预测任务，模型会将  $C^{(L)}$  的池化输出（pooled output）输入到一个线性分类器中，用于预测两段句子是否具有连续关系。具体来说，池化操作会提取  $C^{(L)}$  中 [CLS] 标记对应的隐藏状态，表示为  $C_{[\text{CLS}]}^{(L)}$ ，然后通过一个线性层映射到二分类的概率空间：

$$P(\text{NSP}|C^{(L)}) = \text{softmax}(\text{Linear}(C_{[\text{CLS}]}^{(L)})) \quad (17)$$

NSP 任务的损失同样采用交叉熵损失函数，公式如下：

$$\mathcal{L}_{NSP} = -\frac{1}{M} \sum_{j=1}^M y_j \log(\hat{y}_j) \quad (18)$$

其中， $M$  表示样本数量， $y_j$  表示第  $j$  个样本的真实标签， $\hat{y}_j$  表示模型预测的概率分布。

最终，预训练阶段的总损失由 MLM 和 NSP 两部分损失相加求和得到：

$$\mathcal{L}_{\text{pretrain}} = \mathcal{L}_{MLM} + \mathcal{L}_{NSP} \quad (19)$$

通过上述预训练任务，模型能够学习到丰富的上下文语义信息和句子间的逻辑关系，为下游任务的微调提供强大的语义表示能力。

### 3.1.4 下游任务头

对于下游 NSP 任务，我们使用了一个简单的 Dropout 层和一个线性层来进行分类。Dropout 层用于防止过拟合，线性层用于将模型的输出映射到二分类的概率空间。具体来说，NSP 任务的输出可以表示为：

$$\begin{aligned} NSP^{(L)} &= \text{Dropout}(C^{(L)}) \\ NSP_{output} &= \text{Linear}(NSP^{(L)}) \end{aligned} \quad (20)$$

其中  $NSP_{output}$  是模型在 NSP 任务上的输出， $C^{(L)}$  是模型最后一层的输出。

为了计算 NSP 任务的损失 (Loss)，我们采用了交叉熵损失函数 (CrossEntropy-Loss)。该损失函数用于衡量模型预测的类别分布与真实标签分布之间的差异。具体来说，假设真实标签为  $labels$ ，则损失的计算公式为：

$$\mathcal{L}_{NSP} = -\frac{1}{N} \sum_{i=1}^N \sum_{j=1}^2 y_{ij} \log(\hat{y}_{ij}) \quad (21)$$

其中： $N$  表示样本数量； $y_{ij}$  表示第  $i$  个样本的真实标签 (one-hot 编码)； $\hat{y}_{ij}$  表示模型预测的第  $i$  个样本属于类别  $j$  的概率； $\log$  表示自然对数。

最终，损失  $\mathcal{L}_{NSP}$  将用于反向传播，以优化模型参数，提高模型在 NSP 任务上的性能。

## 3.2 实验准备

### 3.2.1 数据集

为验证本文提出的融合式中文编码模型在真实领域场景下的有效性，我们构建了一个面向政策文本理解的中文语料数据集。该数据集聚焦于中国语境下的政府文件与政策文本，具备鲜明的领域特征和丰富的语言结构，是评估模型在政策解读、政务问答等任务中的理想语料基础。

本研究所用政策文本数据集由网络爬虫技术自动采集所得，涵盖了中国大陆多个省市自治区政府官方网站上发布的各类政策文件、政府工作报告、发展规划、法规条例、政策解读书籍摘要等正式文本资料。具体爬取来源包括但不限于各地政府门户网站（如北京、上海、浙江、四川等）、国家发展改革委员会、财政部、教育部等中央部委官网、政策解读平台与政务服务平台发布的权威二次文本。所有文本均为公开发布内容。

最终构建的政策文本语料共计 99 篇文档，涵盖不同政策类别与区域来源，包括宏观经济、财政税收、乡村振兴、教育改革、公共卫生等多个细分领域。文本总句子

数为 14,652 个句子，总字符数达约 38 万汉字，句子平均长度为 26.1 字/句，具备典型政策文风特征（长句、嵌套多、名词短语密集）。

### 3.2.2 数据预处理

在正式输入模型训练与评估之前，需对原始政策文本数据进行系统性预处理，以确保语料干净、结构统一，并充分提取粒度丰富的语言单元以用于编码器输入。由于部分政策文本来自于扫描版文档或 PDF 转换结果，存在典型的 OCR 噪声，如断行错误、多余换行符、标点残缺、非正文符号嵌入等。我们对文本进行了如下处理：

将所有非段落边界的换行符（`\n`）替换为空格，避免长句被错误截断；修复常见中英文混排标点错误（如“：”变为“:”），统一中文全角标点风格；移除低频率出现的非 Unicode 汉字字符、控制符（如 `\x0c`）、多余空格或乱码；清洗后文本结构更加规整，显著提升了后续“词级信息”提取的质量与鲁棒性。

政策类文本中，常包含表格信息、超链接、页脚编号、批注引用等非自然语言结构。这些信息对于模型训练而言往往是噪声，需予以剔除。本文采用基于正则表达式的规则方法进行清理，匹配带有明显网格、对齐线性结构的文本片段，如“|xxx|yyy|”或“——”连接的列名，并删除。去除所有形如“http[s]”、“.com”的 URL 链接；利用常见模式（如“第 x 页”、“附件 x”、“×× 年 ×× 号”）识别页脚、页码信息，并从段落中剥离；去除括号中的编号批注（如“(1)”、“[图 1]”），并重构自然语言句式。

基于 n-gram 的分词策略在中文文本处理中存在一定局限性，这是因为其通过计算每两个字符中间的 PMI（Pointwise Mutual Information）来确定 n-gram 片段的边界，可能导致切分出一些不符合中文语言习惯的片段，尤其是在处理多义词、成语或专有名词时。此外，n-gram 方法在处理长文本时可能会产生大量的冗余片段，增加了模型的计算复杂度和内存消耗。并且，由于没有引入停用词过滤机制，可能会导致一些常见但对语义理解没有实际贡献的词被纳入模型训练中，从而影响模型的性能。

相比基于 n-gram 的策略，本文使用的 HanLP 提供了更语义驱动的中文分词工具，其基于神经网络词法分析模型，并支持自定义词典增强分词效果。使用 HanLP 提供的 api 接口对清洗后的句子进行切词，去除停用词与其他不符合规则的词，统计词表中的词频，剔除低频词（出现次数小于 10 次）以减少噪声对模型训练的影响。最终构建的词表包含 2662 个词，覆盖了 99.8% 的句子。

### 3.2.3 微调数据集构建

为评估改进后的中文文本编码器对上下文建模能力的增强效果，本文构建了一个适用于中文领域政策文本的 \*\*Next Sentence Prediction (NSP)\*\* 任务数据集。不同于英文 NSP 任务中基于自然段的句子抽取，中文文本中 \*\*句子划分粒度模糊、标点使用灵活\*\*，因此我们结合句号（“。”）与逗号（“，”）进行多粒度分割，设计出更贴合中文语言结构的样本生成策略。

正样本的构建遵循“语义连续、上下文自然”的原则。我们首先使用 HanLP 对原始政策语料进行句子划分，按“中文句号（。）”作为分句边界，得到句子集合：

$$\mathcal{S} = \{s_1, s_2, \dots, s_n\} \quad (22)$$

对于每个句子我们利用“逗号”划分子句：

$$s_i = \{s'_{i,1}, s'_{i,2}, \dots, s'_{i,k}\} \quad (23)$$

将相邻的子句对  $(s_{i,k}, s_{i,k+1})$  作为正样本对  $(s_i, s_{i+1})$ ，即：

$$(s_{i,k}, s_{i,k+1}) \in \mathcal{P} \quad (24)$$

最终得到的正样本对数量为：

$$|\mathcal{P}| = 23374 \quad (25)$$

针对负样本，我们设计了两种生成策略，分别体现不同的训练目标与数据分布假设。

方案一：正负样本比例 1:1，包含“句子顺序扰动”样本

该策略下，负样本数与正样本数相等，即：

$$|\mathcal{N}_1| = |\mathcal{P}| = 23374 \quad (26)$$

在构造负样本  $(s'_i, s'_j) \in \mathcal{N}_1$  时，采用如下两类样本来源：

1. 顺序颠倒型（20%）：从正样本中选取 20% 的句子对  $(s_i, s_{i+1})$ ，反转顺序构造  $(s_{i+1}, s_i)$ ；

2. 跨文档随机型（80%）：从整个语料库中随机选取两个不相邻句子  $(s'_i, s'_j)$ ，其中  $s'_i$  和  $s'_j$  可来自不同文章或不同段落，确保语义无关。

该策略优点在于正负样本数量平衡，有助于训练模型识别语义连贯性，并关注句子顺序是否合理。我们将该方案的数据集称为 PolicySM-1 (Policy Sentence Matching

1)。然而，由于来自不同文章的句子有概率在语义上具有相似性，如两篇介绍地区“数据治理政策”的文章可能都提到“数据治理”这一概念，因此该方案的负样本可能会引入一些噪声，导致模型的训练结果无法很好地反映句子间的真实关系。为此，我们引入了第二个数据集方案。

方案二：正负样本比例 1:5，基于篇内语义混淆构造

在第二种策略中，我们设计更复杂的负样本：从同一篇文章中抽取相隔 2 至 5 句话的句子对，并在其内部使用逗号划分子句以混淆语义边界。即，对于句子序列  $\{s_i\}$ ，构造如下样本对：

$$(s_{i,k}, s_{j,l}) \quad \text{其中} \quad j = i + \delta, \quad \delta \in \{2, 3, 4, 5\} \quad (27)$$

其中  $s_{i,k}$  和  $s_{j,l}$  分别为句子  $s_i$  和  $s_j$  中用逗号划分出的任一子句。最终采样形成：

$$|\mathcal{N}_2| = 5 \times |\mathcal{P}| = 116870 \quad (28)$$

该方案的设计意图在于强化模型对篇内局部干扰与语义跳跃的辨别能力，提升编码器对更细粒度上下文关系的建模能力。尤其是在 RAG 任务中，模型需要能够区分相关与不相关的句子对，以便在检索阶段准确匹配问题与答案。而同一篇文章中相隔较远的句子往往比起随机选取句子更能体现语义上的差异。而更多的负样本数量也有助于模型学习到更丰富的语义信息，避免过拟合。因此，方案二的负样本构造更贴近实际应用场景，能够有效提升模型的泛化能力。我们将方案二的数据集称为 PolicySM-2 (Policy Sentence Matching 2)。

我们分别采用方案一与方案二构建了两个 NSP 训练数据集，分布如表 1 所示。

### 3.2.4 骨干模型

有关于模型的具体实现，本文基于 *HuggingFace* 的 *Transformers* 库，使用了 *ZEN-pretrain-base* 模型和 *bge-base-zh-v1.5* 模型。前者是基于 BERT 的中文预训练模型，拥有预训练的字符嵌入与词嵌入参数。后者是 BAAI 提出的中文通用嵌入模型，主要用于 RAG 任务中的文本嵌入生成。我们将 *bge-base-zh-v1.5* 的主体 BERT 部分参数提取出来作为骨干字符嵌入模型，其余的参数则使用 *ZEN-pretrain-base* 模型进行初始化。微调过的 *bge-base-zh-v1.5* 模型将被用于后续的 RAG 系统中。



数据集	正样本数	负样本数	正负比例	总样本数
PolicySM-1	23374	23374	1:1	46748
PolicySM-2	23374	116870	1:5	140244

表 1: NSP 任务样本分布

在后续实验部分，我们将分别基于两种方案训练 NSP 任务模型，并评估不同样本构造策略对文本编码器语义理解能力的影响。

### 3.2.5 训练配置

实验在一台高性能 GPU 服务器上进行，配置如下：

- GPU: NVIDIA RTX 4090  $\times$  1, 显存 24 GB
- CPU: 20 核 Intel Xeon 处理器
- 内存: 80 GB
- 工作空间: 50 GB SSD 存储用于中间缓存与模型权重
- 框架: PyTorch 2.0, python 3.8, cuda 11.8

该配置确保了在处理大规模预训练模型和多层融合机制时具备稳定的训练性能和内存调度能力。

## 3.3 实验结果

### 3.3.1 PolicySM-1

我们在 PolicySM-1 数据集上测试了不同融合方法与 Hanlp 分词的效果。在这部分实验中，我们仅使用 ZEN-pretrain-base 模型作为基础模型，使用了不同的融合方法（门控机制、多头注意力机制）与原始论文中直接相加进行对比。并对比了使用 Hanlp 分词与使用 n-gram 表示的效果。

预训练和下游任务的训练超参数分别如表 2 和表 3 所示。

参数名称	参数取值	描述
epochs	3	总训练轮数
train_batch_size	32	训练批量
learning_rate	3e-5	初始学习率
warmup_proportion	0.9	学习率预热比例

表 2: PolicySM-1 预训练超参数

参数名称	参数取值	描述
max_seq_length	128	最大输入长度
train_batch_size	32	训练批量
eval_batch_size	8	评估的批量大小
learning_rate	5e-5	初始学习率
num_train_epochs	3	总训练轮数
warmup_proportion	0.9	学习率预热比例

表 3: PolicySM-1 下游任务超参数

实验结果由表 4 所示。可以看到，使用 Hanlp 分词的模型在所有融合方法中表现最好，达到了 0.9132 的准确率。使用 n-gram 表示的模型在 Attention-based Fusion 下也取得了不错的效果，达到了 0.9121 的准确率。这意味着，使用 Hanlp 分词的模型在处理中文政策文本时，能够更好地捕捉到词语之间的关系和上下文信息，从而提高了模型的性能。而两种融合方法与原始论文中直接相加的效果相比，均有了显著提升，说明引入门控机制和多头注意力机制能够有效增强模型对中文政策文本的建模能力。

融合方法	分词方法	准确率
None	n-gram	0.8867
Gated Fusion	n-gram	0.8959
Attention-based Fusion	n-gram	<b>0.9121</b>
None	Hanlp	0.9035
Gated Fusion	Hanlp	0.8981
Attention-based Fusion	Hanlp	<b>0.9132</b>

表 4: PolicySM-1 下游任务结果

### 3.3.2 PolicySM-2

我们在 PolicySM-1 数据集上同时测试了两个模型 (ZEN-pretrain-base 和 bge-base-zh-v1.5) 在不同融合方法下的效果。预训练和下游任务的训练超参数分别如表 5 和表 6 所示。由于先前的实验已经验证了使用 Hanlp 分词的模型在所有融合方法中表现最好, 因此我们在这部分实验中仅使用了 Hanlp 分词。

参数名称	参数取值	描述
epochs	5	总训练轮数
train_batch_size	32	训练批量
learning_rate	3e-5	初始学习率
warmup_proportion	0.9	学习率预热比例

表 5: PolicySM-2 预训练超参数

参数名称	参数取值	描述
max_seq_length	128	最大输入长度
train_batch_size	32	训练批量
eval_batch_size	8	评估的批量大小
learning_rate	5e-5	初始学习率
num_train_epochs	3	总训练轮数
warmup_proportion	0.9	学习率预热比例

表 6: PolicySM-2 下游任务超参数

实验结果由表 7 和表 8 所示。我们可以看到，在预训练任务中，使用了门控机制和多头注意力机制的模型在 MLM Loss、MLM 准确率和困惑度上均有显著提升，尤其是使用多头注意力机制的模型在所有指标上均达到了最优值。这表明，融合式中文编码模型能够有效地捕捉到字符和词之间的关系，从而提高了模型的性能。

而在下游任务中，使用了门控机制和多头注意力机制的模型在 NSP 准确率上也有显著提升，尤其是使用多头注意力机制的模型在 NSP 准确率上达到了最优值。这表明，融合式中文编码模型能够有效地捕捉到上下文信息，从而提高了模型的性能。

骨干模型	融合方法	MLM Loss	MLM acc	困惑度
BGE	none	1.0151	0.7837	2.76
	gate	0.9763	0.7924	2.65
	attn	<b>0.9218</b>	<b>0.8029</b>	<b>2.51</b>
ZEN	none	0.9064	0.8036	2.48
	gate	0.9332	0.7995	2.54
	attn	<b>0.8952</b>	<b>0.8059</b>	<b>2.45</b>

表 7: PolicySM-2 预训练结果

骨干模型	融合方法	NSP acc
BGE	none	0.9236
	gate	<b>0.9311</b>
	attn	0.9272
ZEN	none	0.9260
	gate	<b>0.9325</b>
	attn	0.9274

表 8: PolicySM-2 下游任务结果

## 4 未名问政



图 1: 未名问政 Logo

### 4.1 系统架构

前文通过在政策文本语料上的微调，显著提升了 *bge-base-zh-v1.5* 模型在子领域内对中文语义的理解与表达能力。在此基础上，本文进一步构建了一个基于检索增强生成 (Retrieval-Augmented Generation, RAG) 机制的智能问答系统，命名为“未名问政”，以提供高效、精准的政策解读与咨询服务，帮助用户快速理解并获取所需政策信息。该系统采用了 Langchain-Chatchat 架构作为整体框架，在保障系统模块化、可扩展性的同时，方便快速集成主流语言模型与检索引擎。系统整体由三部分组成：检索模块、生成模块，以及一个轻量级的融合控制层，形成了完整的“查询—检索—生成—响应”工作链条。

在检索模块方面，系统使用了基于 PolicyBERT 架构并经过领域微调的 *bge-base-zh-v1.5* 模型作为中文文本编码器。为进一步融合通用语义能力与子领域知识，本文将微调后的参数与原始模型部分参数拼接，构建了 *bge-base-zh-v1.5-merge* 模型。具体流程为：首先对政策语料进行分段预处理（每段限制在 750 字以内），随后将其编码为固定维度的向量，存入本地 Faiss 向量数据库中。在用户输入查询后，系统会将查询文本同样编码为向量，并通过相似度检索选出 top-k 个最相关的政策片段，作为后续生成模块的上下文支持。

在生成模块中，系统部署了通过 Ollama 实现本地化的 *Qwen2.5 : 7b* 大语言模型。该模型具备长上下文处理能力与良好的中文语义生成效果，能够结合检索到的文本片段和用户原始问题，通过预设提示模板 (Prompt) 完成专业、连贯的答案生成过程。生成模块将检索模块输出的多个相关片段与查询拼接，并输入至 *Qwen2.5 : 7b*，最终输出针对性强、逻辑清晰的回答内容或政策摘要。

综上，本文构建的“未名问政”系统在本地硬件 (RTX 4060 8GB 显存) 上完成了从文本检索到问答生成的全流程部署，充分融合了 *bge-base-zh-v1.5-merge* 的检索能力与 *Qwen2.5 : 7b* 的生成优势，依托 Langchain-Chatchat 架构实现了高度模块化与可配置的中文 RAG 系统。该系统具备部署灵活、响应快速、扩展便捷等特点，尤其

适用于政务智能问答、政策知识库构建与自动解读等典型场景。

## 4.2 系统演示

如图2所示，未名问政-PolicyBert 系统的主界面简洁直观，当前版本为 1.0.0.1。界面左侧设有功能导航栏，用户可在“多功能对话”、“RAG 对话”与“知识库管理”等主要模块间进行切换。在“RAG 对话”模式下，用户可以根据具体需求，自定义相关配置参数。首先，用户可通过下拉菜单选择对话模式，目前支持“知识库问答”；随后可选择已加载的知识库，以支持特定领域的问答服务。进一步地，系统允许用户设置历史对话轮数（默认为 3 轮）、匹配知识条数（默认为 5 条），以及知识匹配的最低分数阈值（以滑动条形式设置，范围为 0.00 至 2.00，默认值为 0.50），SCORE 越小，相关度越高，取到 2 相当于不筛选。此外，用户还可以勾选“仅返回检索结果”选项，以查看未融合生成的原始检索片段。界面右侧为空白对话窗口区域，用于展示系统响应内容。底部提供输入框，支持多轮对话交互，用户输入内容后可按 **Shift+Enter** 执行提交，界面整体设计清晰高效，便于快速部署和应用。

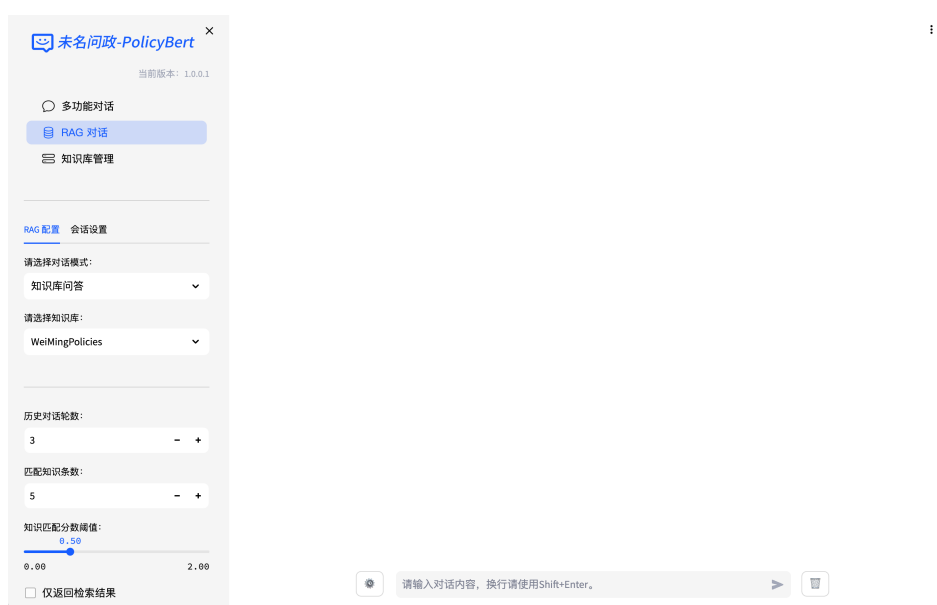


图 2: 未名问政 RAG 系统首页界面。界面左侧为参数配置区域，支持设置对话模式、知识库来源、历史轮数、匹配条数及知识匹配分数阈值，右侧为对话窗口区域，用于显示模型响应结果。该系统通过图形化界面提供了灵活高效的 RAG 任务配置方式。

## 5 结论

数据科学技术为金融风控提供了强大支持。未来，结合更丰富的数据源与更先进的深度模型，有望进一步提升预测能力。

## 参考文献

- [1] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523, 1988.
- [2] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3(null):993–1022, March 2003.
- [3] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.
- [4] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.
- [5] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In Marilyn Walker, Heng Ji, and Amanda Stent, editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.
- [6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.
- [7] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.



- [8] Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, and Hua Wu. Ernie: Enhanced representation through knowledge integration, 2019.
- [9] Shizhe Diao, Jiaxin Bai, Yan Song, Tong Zhang, and Yonggang Wang. Zen: Pre-training chinese text encoder enhanced by n-gram representations. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4729–4740, Online, November 2020.
- [10] Shizhe Diao, Ruijia Xu, Hongjin Su, Yilei Jiang, Yan Song, and Tong Zhang. Taming pre-trained language models with n-gram representations for low-resource domain adaptation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3336–3349, Online, August 2021. Association for Computational Linguistics.
- [11] Robert Lakatos, Peter Pollner, Andras Hajdu, and Tamas Joo. Investigating the performance of retrieval-augmented generation and fine-tuning for the development of ai-driven knowledge-based systems, 2024.
- [12] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.
- [13] Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. C-pack: Packaged resources to advance general chinese embedding, 2023.
- [14] Qian Liu, Jinke Song, Zhiguo Huang, Yuxuan Zhang, glide the, and li-unix4odoo. langchain-chatchat. <https://github.com/chatchat-space/Langchain-Chatchat>, 2024.

- [15] Kaixin Ma, Meiling Liu, Tiejun Zhao, Jiyun Zhou, and Yang Yu. Ptwa: Pre-training with word attention for chinese named entity recognition. *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2021.
- [16] Wenbiao Li, Rui Sun, and Yunfang Wu. Exploiting word semantics to enrich character representations of chinese pre-trained models. In *Natural Language Processing and Chinese Computing*, 2022.
- [17] Qiang He, Guowei Chen, Wenchao Song, and Pengzhou Zhang. Prompt-based word-level information injection bert for chinese named entity recognition. *Applied Sciences*, 2023.