论文题目：基于 PLM-FCNN/CNN 的问答 AIGC 检测

作者：罗维锋、占可欣、皮圣坤

2024/7/26

摘要

AI 大模型滥用现象导致 AIGC 检测成为关注焦点。本文聚焦于知识问答领域的检测问题，基于选定的数据集，建立了合适的上下游网络完成 AIGC 检测，并将不同算法进行比较，对 AIGC 检测提出见解。

数据预处理阶段，首先通过将 HC3 数据集中的问题输入**三种主流中文 AI 大模型**来进行**文本增强**，随后对文本数据进行了**删除不必要词**、**删除所有标点**、**串联问题和答案**和**删除停用词**等等处理。我们使用**选择性文本增强(STA)**的**词角色划分**方法计算两类文本中所有词的**加权对数似然比(wllr)**和**余弦相似度**，从**统计相关**和**语义相关**两个角度来筛选停用词，最终将文本中的 trivial 类和 venture 类词删除。依次对数据集采用了不同的预处理方式，用于后续模型对不同预处理敏感度的对比。

网络搭建的上游任务中，我们采用了**基于 word2vec 的固定编码**和**基于 transformer 上下文编码**两种方式获得文本嵌入。前者需搭配如 DPCNN、TextCNN 等网络来进行文本特征提取以得到文本嵌入，对于词向量的编码缺少上下文联系。而基于 transformer 的**预训练语言模型(PLM)**如 BERT 由于其注意力机制能够考虑整个序列的信息，更好地考虑语义关系。我们最终选定使用 PLM 进行文本嵌入，将固定编码形式作为我们的模型对比。

我们构建了 **FCNN** 与 **CNN** 两种分类器来完成下游分类任务，同时引入**批量归一化**、**Dropout**、**L2 正则化**措施控制过拟合风险。在完成超参数调优后，两种分类器在文本预处理方法 1 上的测试集准确率分别达到 **95.73%**、**95.63%**。我们比较了不同文本预处理方法与分类器组合检测效果，结果表明，分类器与预处理策略互动显著，简单去除特殊符号效果最佳，去除标点或标志词可能对某些分类器产生负面影响。相比于 DPCNN 与 TextCNN，我们的检测器利用基于 Transfomer 的 PLM 能更好地捕获上下文信息，在本检测任务上的表现均超过前两者。最后进行了消融实验，结果表明，我们的检测器各个组件的设计是有效的。

为研究肉眼 AIGC 检测的可能性，在浏览数据集的基础上，我们从**篇幅**、**词汇**、**词性**、**标记词**等方面为两类文本的特征寻找了一些统计依据。最终发现，AI 平均回答长度比人类回答高 **4.25%**，但人类回答中使用词汇量是 AI 的 **1.72** 倍；AI 回答多使用名词、动词、连词，体现出**客观性**和**信息性**，相反，人类回答中多用形容词，更具**主观性**；gold

类词具有高统计性和高语义性，人类回答中多使用语气词而 AI 回答中多使用一些表明严谨性的词汇，这些可以作为二者区分的**指示词**。

最后，根据前文的研究对 AIGC 提出了一些我们的见解。

关键词：AIGC，transformer，word2vec，PLM，CNN

一、 问题重述

1.1 问题背景

2022 年底，ChatGPT 的问世受到学术界和工业界的广泛关注。其诞生使得人工智能生成内容(Artificial Intelligence Generated Content, AIGC)成为新的研究热点。目前国内外涌现出 GPT-4、Claude、New Bing、文心一言、通义千问、星火认知、Kimi 等 AI 大模型，在语言处理、文本生成、知识问答、代码编写、图像处理、长文本处理等领域得到广泛应用。但也存在一些滥用 AI 大模型的现象，如用 AI 写作工具代写论文等，根据《中华人民共和国学位法（草案）》的规定，一旦发现存在人工智能代写等学术不端行为，学位授予单位有权撤销学位。当前，对 AIGC 检测的研究已成为关注焦点。

1.2 问题要求

（1） 选择一个特定领域，收集数据，设计算法，完成 AIGC 检测；
（2） 将你们的检测算法和目前常用的 AIGC 检测算法进行比较；
（3） 能否通过传统方法用肉眼识别 AIGC 呢？
（4） 如何有效进行 AIGC 检测，你们有什么建议？

二、 问题分析

2.1 问题一的分析

AIGC 检测，需要判断一个内容是人类生成的还是 AI 生成的，本质上是一个有监督的二分类问题，即对于输入的内容分类为“人类”和“AI”两个类别。首先需要选定语言处理、文本生成、知识问答、代码编写、图像处理、长文本处理等中的一个领域，本文聚焦知识问答领域的 AIGC 内容检测。

为完成检测，首先需要寻找一个合适的数据集用于模型训练，由于目前的 AI 大模型众多，数据集需要考虑到不同的 AI 模型以增强检测模型的泛化能力，同时考虑合适的预处理方式。文本内容不被机器所理解，因此需要寻找合适的方法来对文本内容进行数字化转换，在此过程中，需要考虑文本的语义特征如何提取。对于文本检测的二分类问题，需要考虑如何去搭建合适的神经网络进行分类。

2.2 问题二的分析

整个分类过程可以分为文本嵌入的上游任务和二分类的下游任务，可以从上下游两个任务角度对不同算法进行对比。

首先需要收集目前常用的 AIGC 检测算法有哪些，上游任务中主要对比不同算法对于文本特征的提取差异；下游任务可以从训练的准确率、训练速度、网络结构等方面进

行对比，同时，由于不同算法对预处理的敏感度不同，可以从预处理的角度对比不同算法对于不同预处理操作的敏感度差异。

2.3 问题三和四的分析

用传统方法肉眼识别，就是要对收集的数据集进行对比分析，从统计意义上获取一些肉眼可观察到的人类文本和 AI 文本的不同，以此作为人眼识别 AI 生成文本的评判依据。根据前文的研究过程得出对 AIGC 检测的建议。

三、 模型假设

- 假设我们的数据来源准确可靠
- 假设数据集中的问题不存在诸如“请用人类的语气回答”的 prompt 语句

四、 符号说明

符号	说明
$AIGC$	人工智能生成内容
$HC3-plus$	数据集
$wllr(w,y)$	加权对数似然比
$similarity(w,l)$	词向量间余弦相似度
$p(w\|y)$	词 w 在类别 y 中出现的概率

五、 数据集分析与文本预处理

5.1 数据集介绍

本论文研究主要关注知识问答领域的短文本 AIGC 内容检测，对于同样的问题，人类和机器分别会如何回应，并检测一个问题是人类还是机器回答。争对本文的研究领域，我们首先需要收集一个比较语料库，其中包含针对相同问题的人类和机器回答，以此作为我们模型的训练基础。

5.1.1 数据集来源

在资料查阅的基础上，我们最终选定 HC3 数据集作为我们的比较语料库。对于中文版的 HC3 数据集，人类回答文本主要来源于两个方面：

1.公开可用的问答数据集，其回答包含人类专家和网络用户的回答

2.维基百科和百度百科构建的问答

HC3 数据集中的机器回答文本主要来源于 ChatGPT，通过将问题输入到 ChatGPT 中获得相应问题的回答。由于目前大语言模型的类别众多，且不同大语言模型在对于问题的回答上具有不同的特征，如国内的智谱清言 GLM-4 在中文对齐能力上整体超过 GPT-4[1]，本文选取了三种中文大语言模型来进行文本增强。选取百度 WebQA 数据集作为对数据集的扩充，将问题输入对应模型的 API 接口来获取相应回答。为防止聊天历史对新问题的回答，我们在 API 接口调用时采用了无状态模式来保证每次会话的独立性。

5.1.2 数据集构成

我们将经过文本增强后的数据集命名为 HC3-plus 数据集，并对于一个问题只选取了一个人类回答。考虑到目前主流大语言模型依然为 ChatGPT，实际 AIGC 检测过程中检测 ChatGPT 模型生成文本概率较大，且本研究受到时间和硬件限制，因此最终对于其他大语言模型扩充比例较小，ChatGPT 回答在数据集中占较大。我们选取的原始 HC3 数据集中的样本涵盖金融、医学、电子、开放问答等多个领域，**表 1** 展示了文本增强后数据集问答样本的构成。

表 1　HC3-plus 数据集样本构成

按领域划分		按 LLM 划分	
Computer science	4617		
Finance	671	ChatGPT	12547
Law	339	文心一言(ERNIE-Speed-128K）	645
Medicine	1061	智谱清言（glm-4）	1694
Nlpcc_dbqa	1672	讯飞星火（Spark4.0Ultra）	817
Phycology	1052		
Oean_qa	6059		

5.2 文本预处理

机器无法直接理解一段文本，在将文本输入分类器前，需要将文本进行词向量编码处理，即将一段文本表示成机器能够处理的向量。原始 HC3 数据集中的数据包括如中文拼音、英文音标、网络符号等无用符号以及常规标点符号，对于文本分类问题，无用符号、字编码、标点符号、停用词等都会对全文语义的理解造成影响，并且影响模型的性能和质量。对于目前的一些预训练模型（PLM），常规的文本预处理也被证实会在一定程度上提高模型的性能。[2]

为了后续研究我们的模型对于不同预处理的敏感度，我们采用了五种不同的文本预处理，如**表 2** 所示。每种处理的详细步骤在后文介绍。

表 2 五种文本预处理

处理	PM1	PM2	PM3	PM4	PM5
删除无关符号	✓	✓	✓	✓	✓
删除所有标点		✓			
结合问题答案			✓		
删除 trivial 类词				✓	
删除 venture 类词					✓

5.2.1 文本清洗与文本输入形式

无关符号包括网络符号，AI 回答中的一些特殊标记符号等等，在实际文本编码中这些无关符号会占用计算资源，且影响了模型编码时对语义的理解等。在预处理阶段我们将所有无关符号去除。

实际研究中发现句子中标点符号的使用会改变句子的语义，而预训练语言模型（PLM）在对文本进行编码过程中会捕捉到标点对句子语义的影响，例如“研究生命的起源”一句话，PLM 对“研究生，命的起源”和“研究，生命的起源”在向量空间中的映射不同。因此我们在 Data processing2 中删除标点用于研究标点对模型分类性能的影响。

对于一般的文本分类检测，我们只需要单独输入人类或机器生成文本就能够完成检测，但对于问答的文本检测来说，研究证实将问题和答案串联输入的 QA 形式通常比单一文本更加稳健[3]，在处理 4 中我们采用将问题和答案串联的方式作为我们模型的输入形式，以此对比在问答类检测中是否输入问题对模型检测的性能影响。

5.2.2 停用词删除

合适的数据处理方式能够提高模型的性能，停用词指的是在文本中大量出现但对文本分类并没有起太大作用的词，对停用词进行删除可用提高模型的计算性能。但在文本分类任务中，一些与文本类别具有高统计相关性的词可能与该类别具有较低的语义相关性，而一些现有的中文停用词库对并未考虑词在不同文本分类任务中的语义相关度[4]，因此本文以语义相关度和统计相关度为依据，采用 STA（Selective Text Augmentation）方法中对词角色的划分，以此进行相应词的删除。

对于本文的文本分类任务，文本对应的标签为$C=\{c_1,c_2\}$，训练集中的所有词集合为V，对于一个词$w_i \in V$和它对应的标签$c_j \in C$，两者之间的统计相似度用加权对数似

然比（*WLLR*）来衡量：

$$wllr(w,y)=p(w|y)\log(\frac{p(w|y)}{p(w|\bar{y})})$$

其中 w 代表分词后词集合中的一个词，y 代表一个分类类别，$\bar{y}$ 表示所有其他的类别。$p(w|y)$ 和 $p(w|\bar{y})$ 分别表示词 w 在该类别中出现的概率。当 w 在 $\bar{y}$ 类别中出现的频率大于 y 时，*WLLR* 值为负。词和标签之间的语义相似度用余弦相似度来衡量：

$$similarity(w,l)=\frac{v_w \bullet v_l}{\| v_w \| \| v_l \|}$$

其中，v_w 和 v_l 分别代表词和标签的向量，使用标签本身的单词或短语即可衡量单词和标签之间的语义相似性，本文两种问答类别的标签分别为“人类的回答”和“人工智能的回答”。

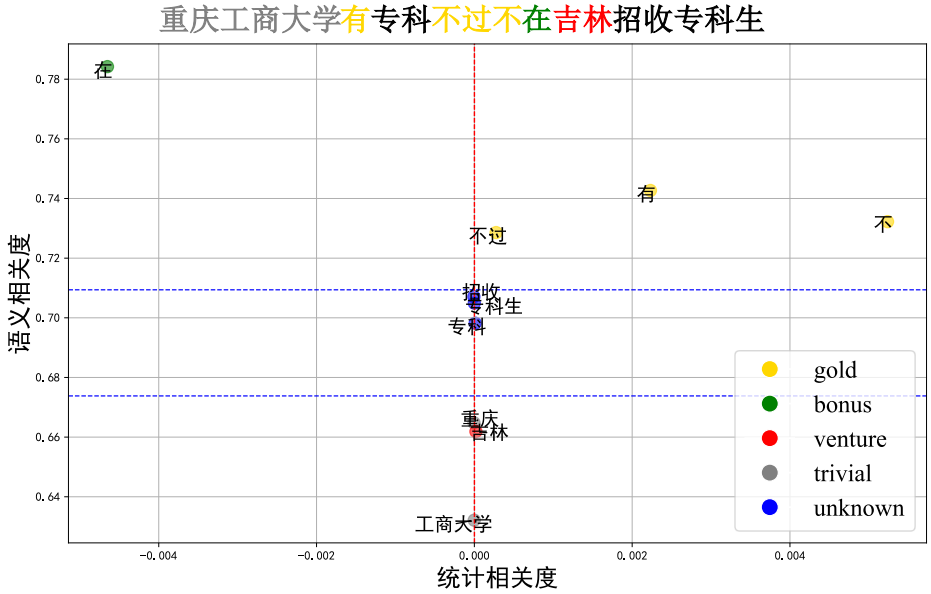


图 1 词角色划分示例（例句：重庆工商大学有专科不过不在吉林招收专科生）

根据两种指标的高低，我们选定两种指标的**上下四分位数**作为高低的阈值，将所有的词划分为四种角色：

- **Gold words**：具有高统计相关性和高语义相关性，属于有用的指示词
- **Venture words**：具有高统计相关性但语义相关性低，极易成为噪声或误导词，不利于模型识别相应的类
- **Bonus words**：具有低统计相关性但语义相关性高，有利于模型的类型判别
- **Trivial words**：具有低统计相关性和低语义相关性，在模型预测中不太重要

在 data processing4 和 data processing5 中，我们分别去除了 trivial 类和 venture 类词，以此检测去除这两类词对于模型的性能的影响。如表 X 所示展示了人类文本中部分词角色分类结果，所有词角色划分结果详见附件：

表 3 词角色示例

类别：人类文本	
Gold word	'毕竟'；'你'；'觉得'；'左右'；'而且'；'就是'；'啊'；'呃'；……
Venture word	'老板'；'孙子'；'吉林'；'君子'；'孙子'；'事半功倍'；……
Bonus word	'喜爱'；'如果'；'身体健康'；'但是'；'是因为'；'这是'；……
Trivial word	'原料'；'水平'；'沈阳'；'工商大学'；'护照'；'胃病'；……

如**表 3** 中所示，“觉得”“呃”等人类常用的口语化词汇通常是评判人类文本的标志性词。面对同一个问题，一些专有名词如“原料”“工商大学”等等，在人类和机器的回答中均会出现，导致统计意义上相关度会较低，对于判断文本类型并无太大用处，而对于数据集中同样的问题，人类文本中更多地使用 venture 类的词汇来进行回答，但这些词与“人类的回答”本身并无太多语义上的联系。因此我们在 data processing4 和 data processing 5 中分别去除 trivial 类和 venture 类的词判断这两类词的去除对于模型性能的影响。

六、 文本嵌入与特征提取

6.1 词嵌入

在将文本输入分类器前首先需要将一段文本转变成机器易于理解的形式，即向量表示。自然语言处理（NLP）中最细的粒度为词，在对文本编码前首先需要对词进行编码。在后续的模型中，我们使用了两类编码形式得到文本嵌入，一些常用的 AIGC 检测算法使用基于 word2vec 的固定嵌入模式，在我们自行搭建的网络中，使用基于 transformer 编码器部分的上下文嵌入模式进行文本嵌入。

6.2 固定嵌入模式——基于 word2vec 的文本嵌入

Word2vec 词嵌入方法将词汇映射到高维空间中的向量，词汇之间的关系通过向量在空间中的距离来表示。传统的 one-hot 编码仅仅将词符号化，并没有体现词和词之间的语义信息，在高维向量中只有一个维度描述了词的信息。Word2vec 模型基于神经网络，利用上下文窗口来预测单词，通过最小化预测的交叉熵损失来获得模型的参数，用来当作输入的某种向量化表示，即词向量。

如图 2 所示，当模型设定窗口为 2，则模型使用当前词的前后各 2 各词来预测当前词。第一层的模型参数W_1与每个输入词汇的 one-hot 编码相乘后即得到每个词的词向量表示，求其平均值作为隐藏层向量的表示，最终输出层用 sigmoid 激活函数表示概率。通过最小化神经网络输出与预测词的真实标签（one-hot）编码，基于损失函数做梯度优

化训练得到对应词汇的词向量编码形式。

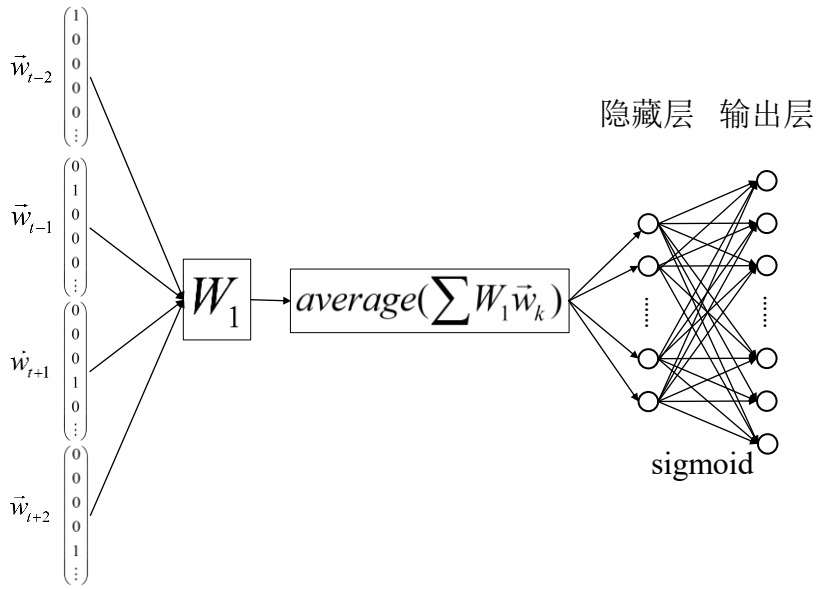


图 2 word2vec-CBOW 原理图（示例窗口为 2）

我们使用 Github 上基于 word2vec 训练的**公开 embedding 词嵌入表**用于后续模型的词向量编码。对于输入的文本，分词后由词嵌入表获得每个词 w_t 对应的词向量 $\vec{x}_t$，随后选用一些 AIGC 检测算法中常用的网络结构如 **DPCNN**[5]、**TextCNN** 等作为**文本特征提取器**，再输入分类器完成文本分类。

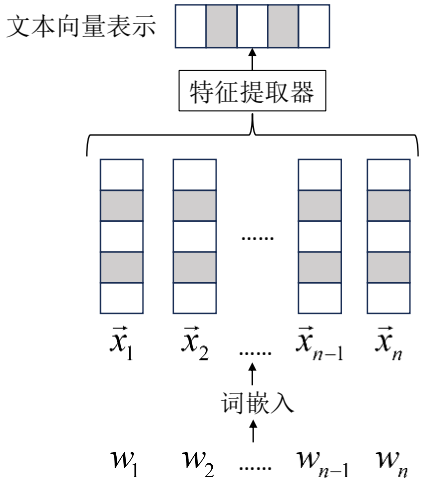


图 3 文本嵌入思路图

6.2 上下文嵌入模式——基于 transformer 的文本嵌入

Word2vec 训练得到的词嵌入表只包含了一个词在不同语境下的一种语义信息，例如，“我喜欢吃苹果”和“我想买苹果手机”两句话中，对于“苹果”这个多义词，word2vec 词嵌入表中的向量表示只包含了其中一种语义信息，这种编码形式无法根据上下文语义来调整词向量。而 **transformer** 由于其**注意力机制**能够更好地捕捉词的上下文语义信息，

因此，我们在后续文本分类的下游任务中，主要采用基于 transformer 的编码方式。

Transformer 通常包括一个编码器模块用于处理输入序列和一个解码器模块用于生成输出序列，在文本嵌入的上游任务中我们主要使用其编码器部分用于获取文本的向量表示。

编码器模块由多个编码器组成，其中包含一系列的自注意力层和前馈神经网络层。注意力机制通过计算查询（Q）和键（K）之间的相似性，得到每个键对应值（V）的权重系数（A），这些权重系数反映了不同信息对当前任务的重要性，由此对该组输入词向量进行更新。编码器帮助理解输入序列的上下文信息，并为每个输入 token 生成一个包含上下文信息的表示。

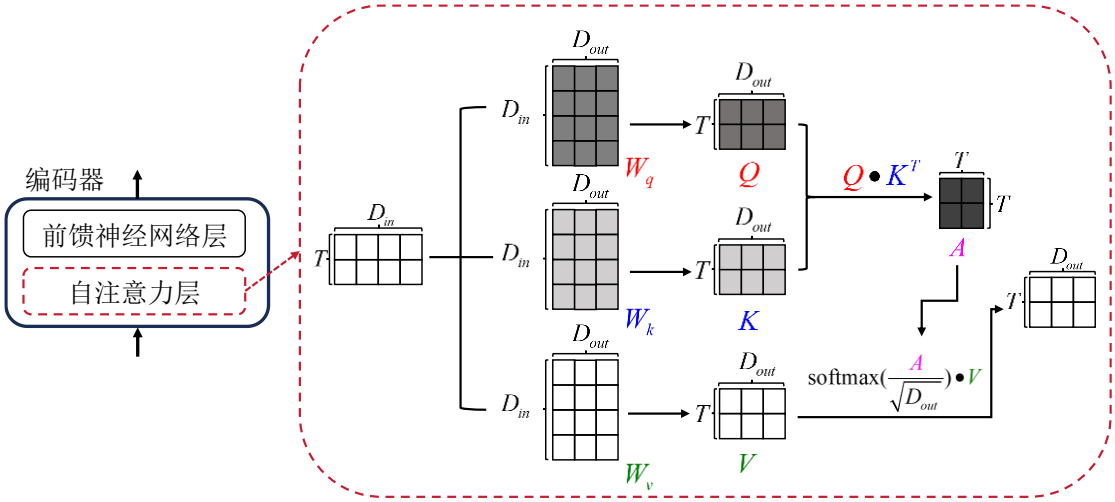


图 4 自注意力机制原理图

我们最终选定 BERT 和 m3e 等预训练的语言模型（PLM）来进行文本编码，选定的 PLM 均在中文语料库上进行训练。对于输入的一段中文文本，PLM 在序列的开始添加[CLS]标记，这些 PLM 会输出包含每个分词后 token 的对应序列，而[CLS]标记对应的输出向量包含了整个输入序列的信息，用作整个序列的表示，并在后续用于我们下游的分类任务。

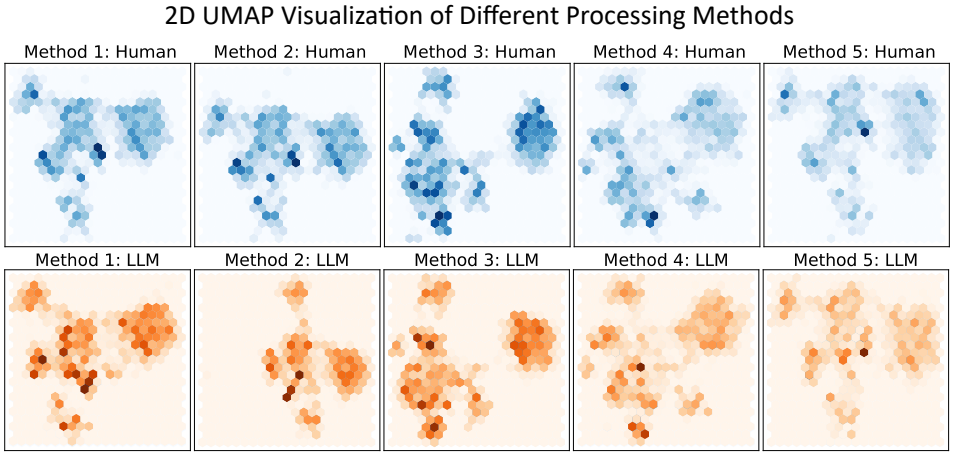


图 5 PLM 编码两类文本向量蜂巢图

使用 PLM 对两类文本进行编码后，上图 UMAP 降维蜂巢图表明，不同预处理方法使得信息结构有了显著变化，但在删减标点符号上（Method1 与 Method2）表现不显著。总的来看，人类文本与 LLM 生成文本的数据特征在二维视图上差异很小，只在局部有所区分，因此需要具备较高性能的分类器将其进一步处理，同时控制过拟合风险。

七、 分类器搭建与结果评估

在完成文本嵌入特征工程之后，将搭建分类器进行有监督学习将文本划分为“人类文本”、“大语言模型生成文本”两类，考虑到文本向量的高维度以及分类复杂性，将搭建神经网络来更好地捕捉文本分类的非线性特征。在训练分类器的过程中，将冻结上游预训练模型参数。在本章中，我们搭建了全连接神经网络以及卷积神经网络两种分类器，同时结合不同的文本预处理方法考察了最终 LLM 检测任务的实现效果。

图 6 上下游模式示意图

7.1 基于全连接神经网络（FCNN）的分类器

7.1.1 网络结构

全连接神经网络（Fully Connected Neural Network, FCNN）是一种最基础的人工神经网络结构。在这种网络中，每一层的每个神经元都与上一层的所有神经元相连接，以及下一层的所有神经元相连接。在本文中，我们设计了一个输入维度为 768 维（与上游 PLM 输出维度保持一致），包含一层 128 维隐含层、一层 64 维隐含层共两层隐含层，输出维度为 1 的全连接神经网络。作为一个二分类的分类器，我们把输出维度设置为 1，以其输出值作为正项的概率，可以较好地避免过拟合问题。同时，将在模型的改进过程中增加批量归一化层。

在 Pytorch 平台中，设置损失函数为 BCEwithLogitloss，其在计算交叉熵损失之前将经过一层 Sigmoid 映射，以此确保计算的稳定与安全，因此在输出层之后便不再添加 Sigmoid 层。本文搭建 FCNN 结构如图 7 所示。

图 7 FCNN 结构示意图

7.1.2 训练与超参数调优

大语言模型生成文本检测是一种特殊的文本分类任务，模型具备大量的参数，我们使用 AdamW 优化器来完成该文本分类任务。AdamW 优化器是 Adam 优化器的一个变种，它在更新规则中直接应用权重衰减（L2 正则化），而在文本分类任务中，权重衰减有助于防止模型过拟合，特别是在处理大量参数的深度学习模型时。

AdamW 优化器具备自适应学习率，能提供较好的收敛速度，但同时，我们设置了 0.01、0.001 两个全局学习率值来寻找较优全局学习率。同时，权重衰减系数也是 AdamW 另一个非常重要的超参数，它将控制模型参数的范围，改变正则化强度，我们设置 0.01 与 0.1 两个梯度。

在文本预处理方法 1 的基础上，通过简单的网格搜索进行超参数调优，可以发现，使用 0.01 学习率后的分类效果普遍降低，而提高权重衰减系数可以平滑损失函数曲线（图 8 ），使得正则化强度更加适应该分类任务的参数复杂性。因此在后续的训练中，将设定学习率为 0.001，L2 权重衰减系数为 0.1。表 X 列举了 FCNN 网络的超参数设定。最后，在每个全连接层前增加批量归一化，以提高数值稳定性。

图 9 汇报了在较优超参数设定下，模型在测试集上的表现，其中，模型在训练集的准确率为 0.9811，在测试集的宏平均准确率为 0.9573、F1 值为 0.9573，有着较好的表现。

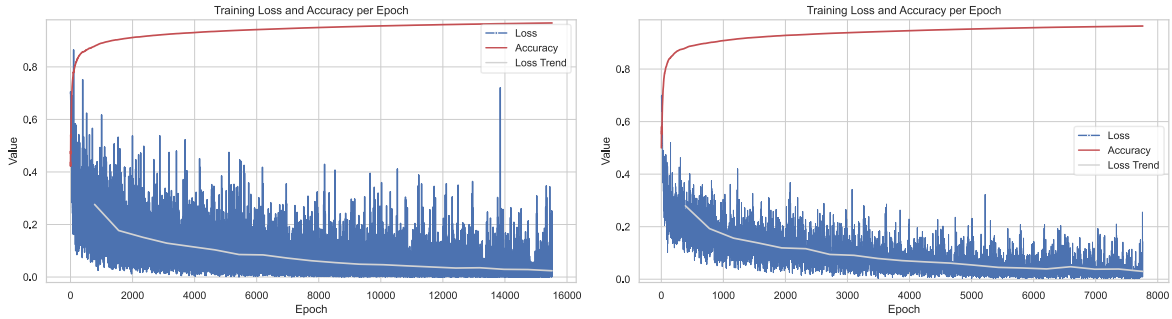
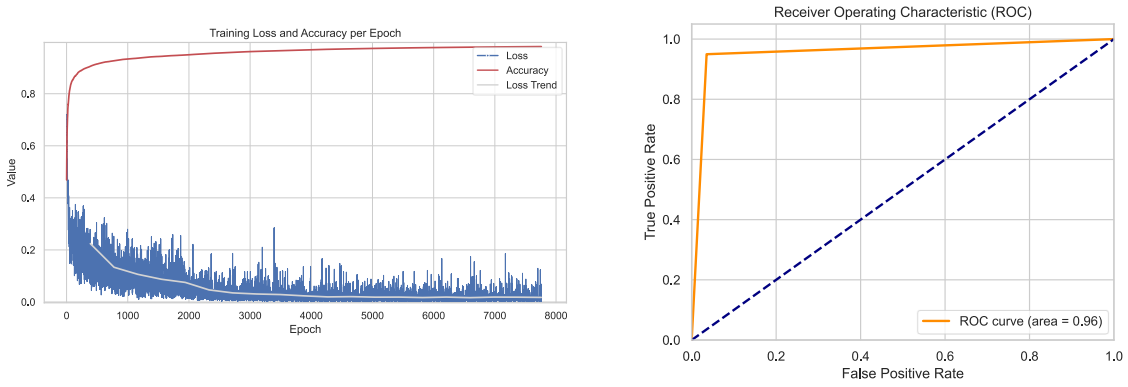


图 8 调整权重衰减系数的训练评估（左：0.001；右：0.01）



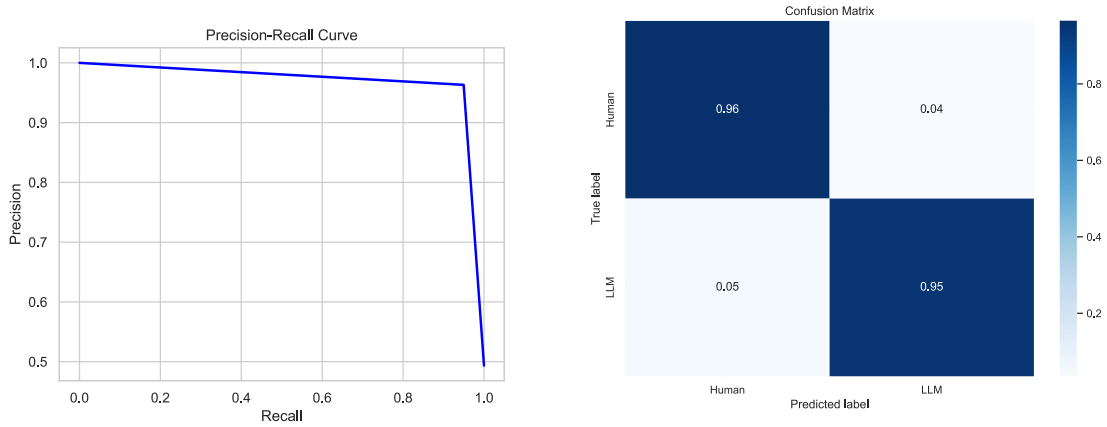


图 9 调优后 FCNN 分类器的测试集评估结果（左上：训练集损失与准确率曲线；右上：测试集 ROC 曲线；左下：测试集 PR 曲线；右下：测试集混淆矩阵）

表 4 FCNN 分类器超参数设定

超参数	值
学习率	0.001
优化器	AdamW
L2 权重衰减系数	0.1
批处理大小	64
训练轮数	20
损失函数	BCE

7.2 基于卷积神经网络（CNN）的分类器

7.2.1 网络结构

卷积神经网络（CNN）具有局部连接和权值共享的特点，同时，通过卷积层，网络可以捕捉到文本中的局部模式，如词组或句法结构，这些模式对于理解文本的语义内容至关重要。我们设计了一个用于下游分类器任务的卷积神经网络，其具有两个卷积池化层以及两个全连接层，并通过在全连接层增加一个 Dropout 层与引入早停机制来控制过拟合。其结构示意图如**图 10** 所示。

卷积层 1 将嵌入向量作为单通道输入，使用 64 个不同的卷积核，每个卷积核的大小为 5；卷积层 2 使用 128 个不同的卷积核，每个卷积核的大小为 5。每个卷积层之后使用 ReLU 激活，并进行批量归一化以平滑数据异常、提高泛化能力，之后进行窗口大小为 2 的最大池化。最后设计两个全连接层，并添加一层 Dropout 层。

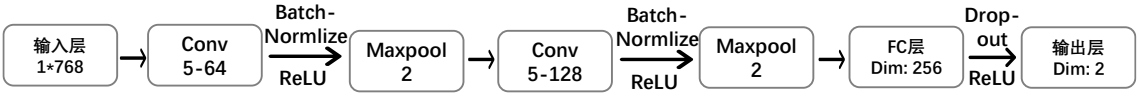


图 10 CNN 结构示意图

7.2.2 训练与超参数调优

基于同样的原因，我们以 AdamW 作为 CNN 分类器的优化器，但考虑到引入了多元的过拟合控制机制，因此将设定较小的权重衰减系数，设置 1e-3、1e-4、1e-5 三个梯度。学习率设置 0.01、0.001 两个梯度。对于 Dropout 的随机丢弃率设置 0.3、0.4、0.5 三个梯度。

在损失函数上，采用 Pytorch 平台下 CrossEntropyLoss 损失函数，其内部包含了两个操作：LogSoftmax 和 NLLLoss（负对数似然损失）。首先，它对输入的 logits 应用 log-softmax 操作，然后计算这些 log-probabilities 和实际标签之间的负对数似然损失，这保证了损失函数对数值稳定性进行了优化，减少了数值下溢的风险。

在文本预处理方法 1 的基础上，经过简单的网格搜索，得到一个较优的结果为设置学习率 0.001、权重衰减系数 1e-4、丢弃率 0.5，其训练与验证评估如图 11 所示。

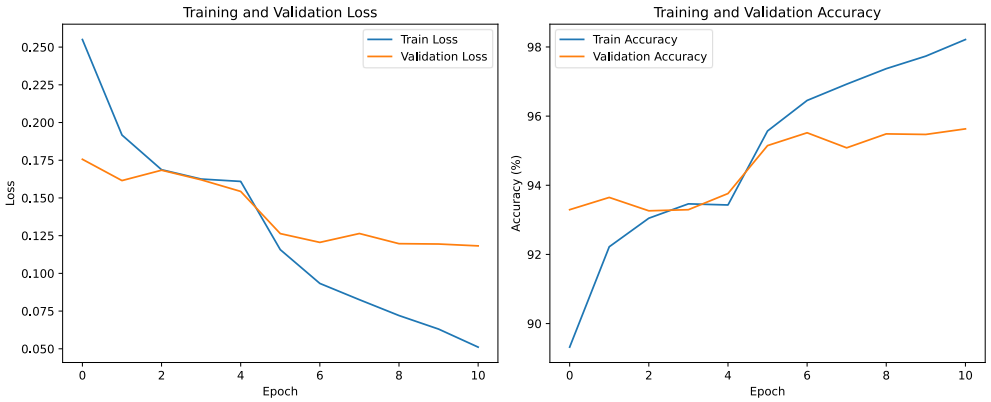


图 11　CNN 训练与验证集评估

图 12 汇报了在较优超参数设定下，模型在测试集上的表现，其中，模型在测试集的宏平均准确率为 0.9563、F1 值为 0.9563，有着较好的表现。

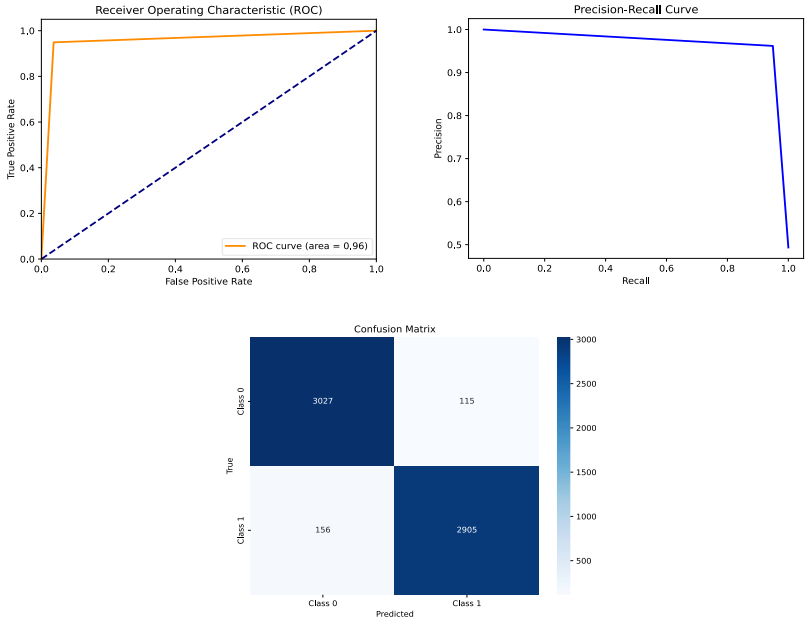


图 12 调优后 CNN 分类器的测试集评估结果

（左上测试集 ROC 曲线；右上：测试集 PR 曲线；下：测试集混淆矩阵）

表 5 CNN 分类器超参数设定

超参数	值
学习率	0.001
优化器	AdamW
L2 权重衰减系数	0.0001
批处理大小	32
训练轮数	20
损失函数	CEL
Dropout 丢弃率	0.5

7.3 LLM 检测任务结果评估

本部分汇报了采用了上文构建的两种分类器并在五种不同的文本预处理方法下对其性能的评估结果，提供准确率、召回率、F1 值三个评估数据，见**表 6** 所示。测试集数据大约 6200 条，且两类数据平衡。

表 6 基于不同文本预处理方法与分类器的 LLM 检测任务的效果评估

分类器	文本预处理方法	Acc	Recall	F1-Score
FCNN	PM 1	**0.9573**	**0.9573**	**0.9573**
	PM 2	0.9368	0.9369	0.9368
	PM 3	0.9544	0.9544	0.9544
	PM 4	0.9515	0.9515	0.9515
	PM 5	0.9550	0.9550	0.9550
CNN	PM 1	**0.9563**	**0.9563**	**0.9563**
	PM 2	0.9310	0.9310	0.9310
	PM 3	0.9478	0.9479	0.9478
	PM 4	0.9434	0.9434	0.9434
	PM 5	0.9381	0.9381	0.9381

基于上述结果可以看到，分类器的选择与预处理策略之间存在一定的相互作用。具体而言，包括：

- 在仅去除特殊符号的预处理方法（PM 1）下，两种分类器的表现均表现出色，准确率、召回率和 F1 分数均接近或超过 **0.95**。这表明，对于 FCNN 和 CNN 而言，简单

的去除特殊符号已经是一个有效的预处理步骤，能够显著提升模型的检测性能。

- 当进一步去除所有标点（PM 2）时，我们发现两种分类器的性能略有下降，尤其是 CNN 的分类效果更为明显。这一结果表明，去除所有标点可能对于某些分类器来说并非最优选择，因为它可能会丢失文本的部分结构信息，从而影响分类效果。
- 在考虑问题与答案联合输入进行嵌入的预处理方法（PM 3）时，我们发现 FCNN 的性能略有提升，而 CNN 的表现则有所下降。这表明联合输入问题与答案的策略可能更适合 FCNN，而对于 CNN 而言，这种预处理方法的效果并不明显。
- 去除 trivial 类词（PM 4）的预处理方法在两种分类器上的表现均略低于 PM 1，但仍然保持较高的性能。这说明去除 trivial 类词对模型性能有一定的影响，但影响程度有限。
- 去除 venture 类词（PM 5）的预处理方法在 FCNN 上的表现略低于 PM 1，而在 CNN 上表现最差。这表明去除 venture 类可能对 CNN 的分类效果产生负面影响，因为它可能会丢失一些关键的语义信息。

上述结果表明，在选择文本预处理方法时，应充分考虑分类器的特点及其对文本信息的依赖程度。在本研究中，FCNN 在大多数预处理方法上的表现优于 CNN，而 PM 1 作为一种简单的预处理策略，在两种分类器上都取得了较好的效果。

接下来，我们针对当前主流的四个大语言模型的数据集分别进行了测试。针对每个大语言模型，设计了 50 条问题，每个问题下均有人类的回答与对应大语言模型的回答，数据集随机打乱，共含有 100 条平衡数据。测试结果如**表 7** 所示。

表 7 针对不同大语言模型的测试评估

分类器	**文本预处理方法**	**Acc**				
		ChatGPT	文心一言	讯飞星火	智谱清言	Avg
FCNN	PM 1	99.00%	81.00%	97.00%	92.00%	92.25%
	PM 2	97.00%	75.00%	87.00%	87.00%	86.50%
	PM 3	93.00%	86.00%	98.00%	96.00%	**93.25%**
	PM 4	96.00%	77.00%	93.00%	94.00%	90.00%
	PM 5	97.00%	80.00%	95.00%	92.00%	91.00%
CNN	PM 1	98.00%	94.00%	96.00%	94.00%	95.50%
	PM 2	96.00%	86.00%	93.00%	86.00%	90.25%
	PM 3	98.00%	98.00%	95.00%	98.00%	**97.25%**
	PM 4	98.00%	93.00%	96.00%	94.00%	95.25%
	PM 5	98.00%	91.00%	97.00%	95.00%	95.25%

考虑到训练数据集中对于四种大语言模型语料的不平衡性（大量的语料来自

ChatGPT 生成），暂不对模型在不同大语言模型间的横评进行分析。从不同的分类器角度来看，FCNN 和 CNN 在不同预处理方法下展现出了不同的性能特点。CNN 在多数预处理方法下的平均准确率高于 FCNN，尤其是在 PM 3 下，CNN 的表现显著优于 FCNN，这表明 CNN 可能更适合处理需要捕捉问题和答案之间复杂关系的数据。然而，这样的结果在**表 7** 中的体现却不够明显，且 FCNN 分类的预测准确率普遍偏低，这可能是因为本次测试样本数量偏少导致了有偏估计，但这值得进一步研究。另外，文心一言在 PM 3 下表现提升，这可能意味着该模型生成的文本在联合问题和答案的上下文中更容易被分类器识别。

八、 模型对比与消融实验

8.1 模型对比

DPCNN（Deep Pyramid Convolutional Neural Network）是一种深度卷积神经网络结构，它在文本分类任务中表现出色，是由 Jin, K.等人于 2017 提出的。DPCNN 通过堆叠多个卷积块来构建一个深度网络，能够捕捉文本中的层次化特征与上下文关系，而没有基于自然语言处理常用的 Transformer 结构。

TextCNN 是一种用于文本分类的卷积神经网络（CNN）架构，它由 Yoon Kim 在 2014 年提出。TextCNN 利用卷积神经网络在处理序列数据方面的优势，特别是自然语言文本，能够有效地提取文本中的局部特征，并进行分类。同样的，TextCNN 基于卷积提取文本的语序与语义特征而非基于 Transformer。

本问所构建的 LLM 检测器均采用基于 Transformer 训练的 PLM，利用注意力机制等捕获上下文信息。本节将基于上文评估时所采用的测试集数据，对于这几种模型的效果进行比较。

表 8 不同检测模型效果对比

架构	文本预处理方法	Acc	Recall	F1-Score
PLM+FCNN	PM 1	**0.9573**	**0.9573**	**0.9573**
	PM 2	0.9368	0.9369	0.9368
	PM 3	0.9544	0.9544	0.9544
	PM 4	0.9515	0.9515	0.9515
	PM 5	0.9550	0.9550	0.9550
PLM+CNN	PM 1	**0.9563**	**0.9563**	**0.9563**
	PM 2	0.9310	0.9310	0.9310

	PM 3	0.9478	0.9479	0.9478
	PM 4	0.9434	0.9434	0.9434
	PM 5	0.9381	0.9381	0.9381
DPCNN	PM 1	0.8616	0.8616	0.8613
	PM 2	0.8362	0.8362	0.8361
	PM 3	0.6077	0.6077	0.5942
	PM 4	**0.8665**	**0.8665**	**0.8663**
	PM 5	0.8654	0.8654	0.8652
TextCNN	PM 1	0.8884	0.8884	0.8884
	PM 2	0.8680	0.8680	0.8680
	PM 3	0.5866	0.5866	0.5786
	PM 4	0.8856	0.8856	0.8856
	PM 5	**0.8901**	**0.8901**	**0.8901**
Bert	PM 1	0.9276	0.9276	0.9276
	PM 2	0.9138	0.9138	0.9137
	PM 3	0.6855	0.6855	0.6710
	PM 4	0.9192	0.9192	0.9192
	PM 5	**0.9336**	**0.9336**	**0.9335**

基于上述结果，我们发现 PLM+FCNN 和 PLM+CNN 在多数预处理方法下均展现出较高的准确率、召回率和 F1 分数，显示出它们在处理文本数据时的强大性能和稳定性。相比之下，其他三种模型在某些预处理方法下表现出色，但整体稳定性不及 PLM+FCNN 和 PLM+CNN。

相对来说，DPCNN 在 PM 1 和 PM 5 下的表现相对较好，但在 PM 3 下性能显著下降，这表明 DPCNN 可能不擅长处理需要联合问题和答案的文本数据。而 TextCNN 在 PM 1 和 PM 5 下的表现较为一致，但在 PM 3 下也出现了性能下降，与 DPCNN 相似，这可能指出 TextCNN 在处理联合数据时的局限性。Bert 模型在 PM 1 和 PM 5 下展现了准确率的提高，这表明 Bert 模型能够有效地利用这些预处理方法来提升性能。然而，在 PM 3 下，Bert 模型的性能显著下降，这可能是由于 Bert 模型的设计并不完全适合处理问题和答案的联合输入。

总的来看，PLM+FCNN 和 PLM+CNN 在多数预处理方法下都优于其他三种模型，显示出它们在处理文本数据时的强大性能和稳定性。表明 PLM+分类器的架构具有一定的优势。

8.2 消融实验

本节将通过消融实验，分析不同组件对 LLM 检测任务性能的影响。首先，我们探究了更换 PLM 对于模型的影响。在文本预处理方法 1 的基础上，采用“m3e-base”、“bge-base-zh-v1.5”、“text2vec-base-chinese”三种针对中文语料进行专门训练的预训练语言模型，结果表明“m3e-base”在 FCNN 分类器与 CNN 分类器中均有更好的表现，因此后续实验将采用“m3e-base”作为 PLM。

接下来，我们将分类器替换为 KNN，旨在探究文本嵌入矩阵本身是否能够有效区分 LLM 文本与人类文本。 其次，将词嵌入向量直接取 Mean pooling 作为文本嵌入以探究 PLM 中上下文处理的影响。测试结果如**表 9** 所示。

表 9 消融实验测试结果

方法	Acc	F1-Score
KNN 分类器	0.7624	0.7598
Max pooling 文本嵌入	0.6832	0.6801

使用 KNN 分类器时，模型的准确率为 0.7624，F1-Score 为 0.7598。这表明，仅使用文本嵌入矩阵本身，KNN 分类器能够在一定程度上区分 LLM 文本与人类文本，但效果不佳，需要使用神经网络分类器进一步处理。当将词嵌入向量直接进行 Max pooling 作为文本嵌入时，模型的准确率为 0.6832，F1-Score 为 0.6801。这表明，相比于使用“m3e-base”预训练语言模型，Max pooling 文本嵌入方法在分类性能上有所下降，这表明上下文理解对于分类性能提升至关重要。

九、 AIGC 肉眼识别与建议

9.1 肉眼 AIGC 识别

神经网络的结构能够提取到输入数据之间的深层特征，但是人眼从文本数据中能够获取的特征有限，初步浏览数据集，我们发现了一些两类文本上的差异，具体体现在回答篇幅、使用词汇量、使用词性、标记词等等上。在第五章工作的基础上，我们统计了数据集中的相关指标，为肉眼识别抓取文本特征寻找一些统计依据。

9.1.1 回答篇幅与词汇量的使用差异

如**表 10** 所示，我们统计了数据集中人类回答文本和 AI 回答文本的文本长度和词汇量数据，从数据结果来看，与 AI 回答相比，人类回答相对较短，但是即使如此，人类所有回答中使用的词汇量却更多。

在面对同一问题时，AI 倾向于更加详细的回答，但从多个问题回答来看，其表达更加单一，相反，人类回答在表达中使用的词汇更加多样化，词汇量远多于 AI 回答。

表 10 文本长度和词汇量统计

	文本长度				词汇量总数
	下四分位数（Q1）	中位数（Q2）	上四分位数（Q3）	均值	
人类文本	60.00	132.00	295.00	239.27	**99009**
AI 文本	**125.00**	**195.00**	**315.00**	**249.46**	57493

9.1.2 词性使用差异

我们使用 jieba 对人类和 AI 回答文本进行了分词和词性标注处理，统计了数据集中两类回答中各种词性出现的频率，如**图 13** 所示。

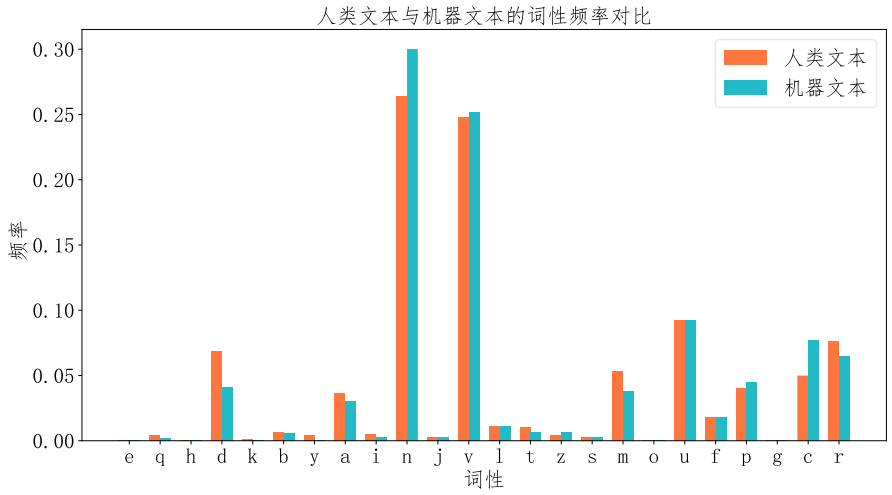


图 13 人类文本和 AI 文本词性使用频率统计（词性均取对应英文单词首字母）

AI 回答中对名词（n）、动词（v）、连词（c）的使用都高于人类。名词的大量使用表明文本更具议论性，体现出了 AI 回答的信息性和客观性。实际浏览 AI 回答过程中发现，AI 更倾向于争对问题给出详尽的回答，在回答时保持客观性和中立性，因此对于专业名词和概念的解释方面更加出色。连词与名词、动词的频繁共现表明文章的结构和因果关系、进展关系或对比关系是清晰的，这也是学术论文或官方文件的典型特征[6]。

相反，人类回答中对于形容词（a）的使用频率高于 AI 回答，可见人类回答更具有主观性，常常使用一些带有主观色彩的词汇。

9.1.3 标记词差异

第五章中我们对两类文本中的词进行了词角色划分。

图 14 展现了词角色分类过程中计算出的 *WLLR* 值和余弦相似度。总体来看，AI 类文本中有部分词和“人工智能的回答”这一类别强相关，其在 AI 类别中被观察到的概率远大于人类文本，如“您”“可能”“或”等词汇，这类词汇很可能成为识别 AI 类文本的标志

词，同时 AI 类文本中也有大量词的 *WLLR* 为负数，说明有大量的词在人类回答中出现的概率远高于 AI 回答文本。

从语义相关性来看，人类回答中的词汇总体和“人类的回答”这一标签语义相关性更高，而 AI 类文本的词汇和“人工智能的回答”之间语义联系较弱。但由于词汇的歧义性，我们认为肉眼难以从语义角度来进行两类文本的识别。

表 11 两类文本中 gold word 部分示例

	Gold word
人类文本	'毕竟'；'你'；'觉得'；'左右'；'而且'；'就是'；'啊'；'呃'；'吧'；……
AI 文本	'和'；'可能'；'或'；'可以'；'通常'；'您'；'信息'；'如果'；'提供'……

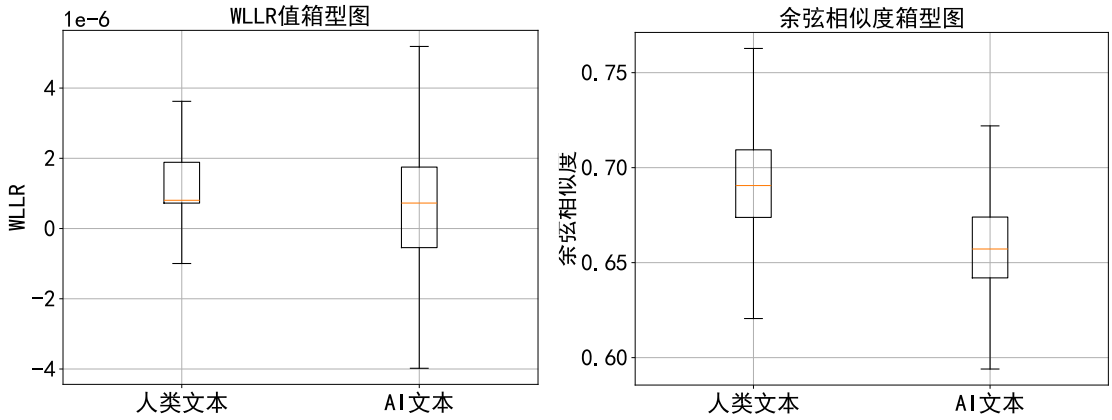


图 14 两类文本 *WLLR* 值和余弦相似度值分布箱型图

在肉眼识别过程中，可以从回答文本中对于标记词的使用频率来进行两类文本的识别，**表 11** 列出了两类文本中部分被划分为 gold word 的词汇。人类回答中常常使用“吧”等语气词，而 AI 回答中对于连词以及“可能”等体现严谨性的词使用较为频繁，这些标记词可以作为肉眼识别时的依据。

总结来看，人类文本和 AI 文本具有以下不同，可以成为肉眼识别 AIGC 内容的依据：

- AI 更倾向于较长的回答，对一个问题提供更加详尽的答案，相反，人类回答更短。
- AI 回答时的表述方式更加单一，人类更倾向使用多样化的表达，表达词汇更加丰富。
- AI 的回答更表现出专家性和客观性，人类更喜欢主观性的表达。
- AI 回答和人类回答各自的标记词可作为识别依据。

在 Chatgpt 的图灵测试中，对于普通大众来说， 用肉眼进行 AIGC 识别，准确率只有大约 50%，而人类专家对于 AIGC 的识别准确率可达到 81%[3]。我们认为在熟知人类回答和 AI 回答差异性的前提下，掌握两类回答的特征，能够用肉眼完成 AIGC 检测，虽然传统的肉眼识别准确率远不及神经网络分类器。

9.2 AIGC 检测建议

对于知识问答领域的 AIGC 内容检测，由于知识问答领域的回答多数为短文本，因此，由前文模型对不同预处理方式的训练结果可知，对文本数据预处理时，保留必要的标点可以使模型更好地捕捉到输入文本的语义信息。停用词的去除，对于模型最终的分类结果并无太大影响，反而能够减少模型的训练时间。因此 AIGC 检测时需要采取合适的预处理方式，合适的预处理方式可以提高 AIGC 检测的精度。

虽然一般的 AIGC 检测模型在我们收集的多领域数据集上总体表现出最高 96%的精确度，远高于 Chatgpt 的图灵测试中人类专家的平均判别率 81%。但在一些专业化程度较高的领域中，例如法学、医学、心理学等领域，人类专家对于 AIGC 内容的判别率几乎达到了 98%。而在这些专业领域，AI 回答可以捏造事实进行回答，例如，在法律领域，AI 回答中可能引用不存在的法律条文。对于知识的捏造，常规的 AIGC 检测算法难以识别。因此未来 AIGC 检测发展方向可能需要将知识真伪识别考虑在内。

参考文献

[1] https://maas.aminer.cn/dev/howuse/glm-4

[2] Siino M, Tinnirello I, La Cascia M. Is text preprocessing still worth the time? A comparative survey on the influence of popular preprocessing methods on Transformers and traditional classifiers[J]. Information Systems, 2024, 121: 102342.

[3] Guo B, Zhang X, Wang Z, et al. How close is chatgpt to human experts? comparison corpus, evaluation, and detection[J]. arxiv preprint arxiv:2301.07597, 2023.

[4] Guo B, Han S, Huang H. Selective text augmentation with word roles for low-resource text classification[J]. arxiv preprint arxiv:2209.01560, 2022.

[5] Johnson R, Zhang T. Deep pyramid convolutional neural networks for text categorization[C]//Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2017: 562-570.

[6] Schleppegrell M J. The language of schooling: A functional linguistics perspective [M]. Routledge, 2004.

附录

由于本文所使用的代码均为项目开发制，涉及文件较多，其中大多数据文件如

json 文件无法在附录中展示，此处只附上附件文件列表。

文件名	所属文件夹	文件类型	说明
data processing1.py	data processing1	Python 文件	第一种数据处理
box_plot_length.py	data processing1	Python 文件	文本长度箱型图绘制
word_character.py	data processing1	Python 文件	词性统计
所有 txt 文件	data processing1	Python 文件	标注了标签的处理后数据
所有 xlsx 文件	data processing1	Python 文件	处理后数据
data processing2.py	data processing2	Python 文件	第二种数据处理
所有 txt 文件	data processing2	txt 文件	标注了标签的处理后数据
所有 xlsx 文件	data processing2	xlsx 文件	处理后数据
data processing3.py	data processing3	Python 文件	第三种数据处理
所有 txt 文件	data processing3	Python 文件	标注了标签的处理后数据
所有 xlsx 文件	data processing3	Python 文件	处理后数据
data processing4.py	data processing4	Python 文件	第四种数据处理
所有 txt 文件	data processing4	txt 文件	标注了标签的处理后数据

所有 xlsx 文件	data processing4	xlsx 文件	处理后数据
data processing5.py	data processing5	Python 文件	第五种数据处理
所有 txt 文件	data processing5	txt 文件	标注了标签的处理后数据
所有 xlsx 文件	data processing5	xlsx 文件	处理后数据
word_freq.py	data processing-role division	Python 文件	词频统计，计算 wllr
word_semantic.py	data processing-role division	Python 文件	语义统计，计算余弦相似度
box_plot_semantic.py	data processing-role division	Python 文件	语义相关度箱型图
box_plot_WLLR.py	data processing-role division	Python 文件	wllr 值箱型图
roles division.py	data processing-role division	Python 文件	划分词角色
所有 json 文件	data processing-role division	json 文件	划分结果，文件均为[词角色]_[类别]方式命名
HC3.py	原始数据和 AI 模型	Python 文件	原始 HC3 数据集读取文件
WebQA.py	原始数据和 AI 模型	Python 文件	原始 WebQA 数据集读取文件
文心一言大模型.py	原始数据和 AI 模型	Python 文件	文心一言模型 API 接口调用
讯飞星火.py	原始数据和 AI 模型	Python 文件	讯飞星火模型 API 接口调用

智谱清言.py	原始数据和AI 模型	Python 文件	智谱清言模型 API 接口调用
\	PLM-CNN	文件夹	PLM-CNN 项目文件
\	PLM-FCNN	文件夹	PLM-FCNN 项目文件