**自然语言处理作业**

不同模型在文本分类中的效果对比

姓 名 ： 张克勤

学 号 ： 2272064

专 业 名 称 ： 计算机技术 (硕)

授 课 老 师 ： 肖桐教授

学 院 ： 计算机科学与工程学院

二○二二年秋季

目录

[一、 作业内容及目的简介 1](#_Toc124345296)

[二、 数据集介绍 1](#_Toc124345297)

[(一) 数据集来源介绍 1](#_Toc124345298)

[(二) 数据集文本内容及文本标签 2](#_Toc124345299)

[(三) 数据集清洗 2](#_Toc124345300)

[三、 不同模型下文本分类效果的对比 2](#_Toc124345301)

[(一) CNN模型结构及在CNN模型下的文本分类实验结果 2](#_Toc124345302)

[1、CNN模型结构 2](#_Toc124345303)

[2、CNN模型代码 3](#_Toc124345304)

[3、CNN模型下文本分类结果 4](#_Toc124345305)

[(二) RNN模型结构及文本分类的实验结果 5](#_Toc124345306)

[1、RNN模型结构 5](#_Toc124345307)

[2、RNN模型代码 6](#_Toc124345308)

[3、RNN模型下文本分类结果 6](#_Toc124345309)

[(三) Transformer模型结构及文本分类的实验结果 7](#_Toc124345310)

[1、Transformer模型结构 7](#_Toc124345311)

[2、Transformer模型代码 8](#_Toc124345312)

[3、Transformer模型下文本分类结果 9](#_Toc124345313)

[(四) 总结与分析 10](#_Toc124345314)

# 作业内容及目的简介

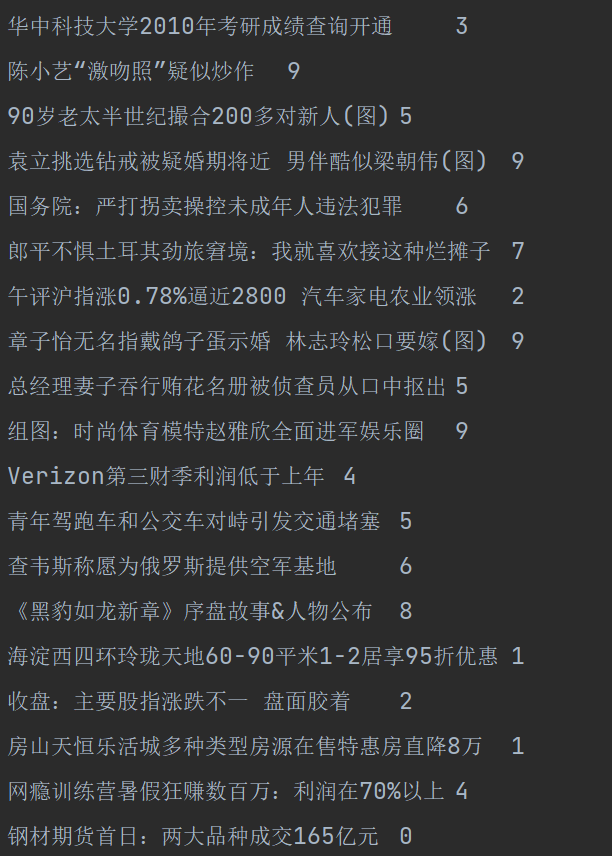
本作业将中文文本分类作为实验任务，依托于新浪新中经过筛选过滤生成文本数据，结合老师上课所讲授的CNN、RNN及Transformer三种模型进行训练，分别实验，完成文本分类的任务。同时本实验以精确率（Precision）、召回率（Recall）和F1值作为实验结果的评价指标，最终将评价结果对比分析，为后续学习奠定基础。

# 数据集介绍

## 数据集来源介绍

本实验根据开源的新浪新闻中数据筛选过滤生成，包含74万篇新闻文档，均为UTF-8纯文本格式。我们在原始新浪新闻分类体系的基础上获取数据集，数据集地址为：http://thuctc.thunlp.org/

数据集中共包含10个不同类别的中文文本，将数据集划分为训练集、验证集和测试集。这里摘取其中十个进行训练，验证和测试。其中训练数据集有180000个，验证数据集和测试数据集各10000个。下图是本次数据集的部分展示，具体如图2-1。

**图2-1 数据集截图**

## 数据集文本内容及文本标签

本次数据集文本中共有十种不用的种类，分别为finance，realty，stocks，education，science，society，politics，sports，game，entertainment十个类别，分别被标注标签0，1，2，3，4，5，6，7，8，9 由上图2-1可知，包含2个字段：文本字段和标签类别字段，每个数字即代表不同的文本类别。

## 数据集清洗

数据清洗是数据预处理的重要部分，在这个部分中，本实验主要完成数据完整性、唯一性的检查：包括但不限于对空值、缺失值，重复文本，文本空格，异常值，非法字符等进行相关检查。同时对数据相关信息进行统计性描述和可视化处理，便于后续实验中对实验效果进行分析。

本数据存在一定的特殊性，由于数据集已经标注好，且其中不存在遗漏标注，数据标签唯一，这为实验的最终完成节省了很多工作。

# 不同模型下文本分类效果的对比

## CNN模型结构及在CNN模型下的文本分类实验结果

### 1、CNN模型结构

卷积神经网络(cnn) 是包含卷积计算且具有深度结构的前馈神经网络。其结构包含了输入层、卷积层、激活函数、池化层、全连接层，即INPUT（输入层）-CONV（卷积层）-RELU（激活函数）-POOL（池化层）-FC（全连接层）。各层在结构中所起作用如下。

1.输入层：数据输入层主要是对原始数据进行初步处理，使卷积神经网络能有更好的效果。常用的处理方法有灰度化和归一化。

2.卷积层：提取输入图片中的信息，这些信息被称为图像特征，这些特征是由图像中的每个像素通过组合或者独立的方式所体现，比如图片的纹理特征，颜色特征。

3.激活函数：激活函数是参考了生物上神经元的机制，在生物意义上的神经元中，只有前面的树突传递的信号大于神经元的阈值时，下一个神经元才会被激活。与之类似，激活函数RELU当输入小于0时，输出为零，即失效；大于零输出则为本身。常见的激活函数有：

**（1）**RELU**函数**

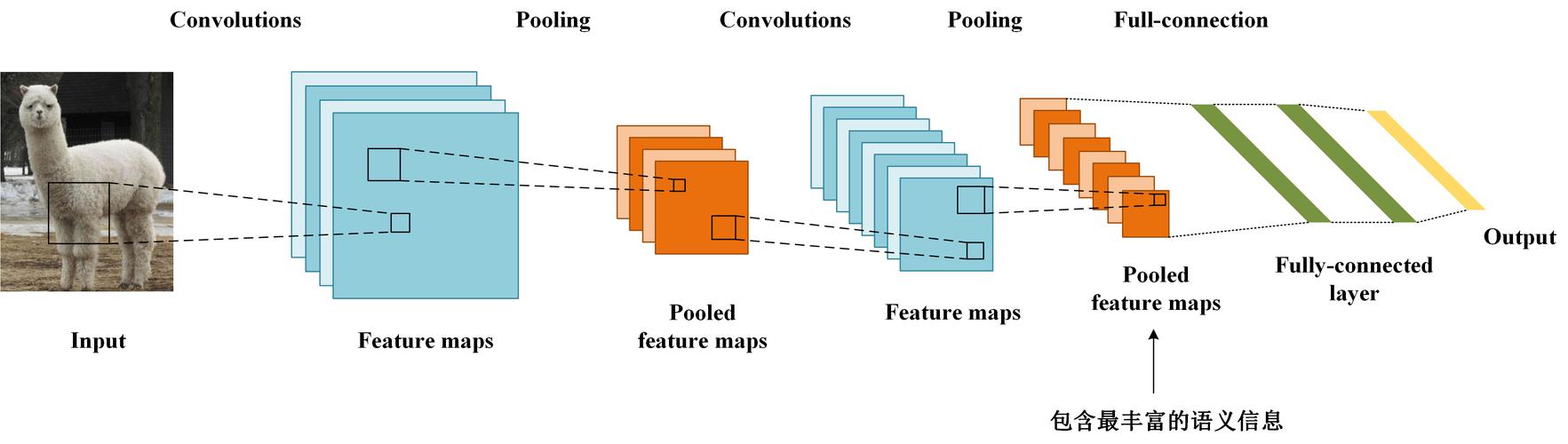
（2）Sigmoid函数

（3）Tanh函数

4.池化层：池化层是对输入的数据进行压缩，提取主要特征的过程。一般池化分为最大池化和一般池化。以最大池化为例，池化的运算跟卷积有些相似，一般以一个2 \* 2的矩阵对输入的数据进行扫描，取2 \* 2区域的最大值。

5.全连接层：全连接层实际上就是传统的神经网络（DNN），每一神经元与输出层的神经元通过权值相连，激活函数一般为softmax函数。当全连接层输入的数据是二维或三维矩阵时，应该拉伸成一维的列表运算。

下图为CNN模型的结构示意图（图3-1）



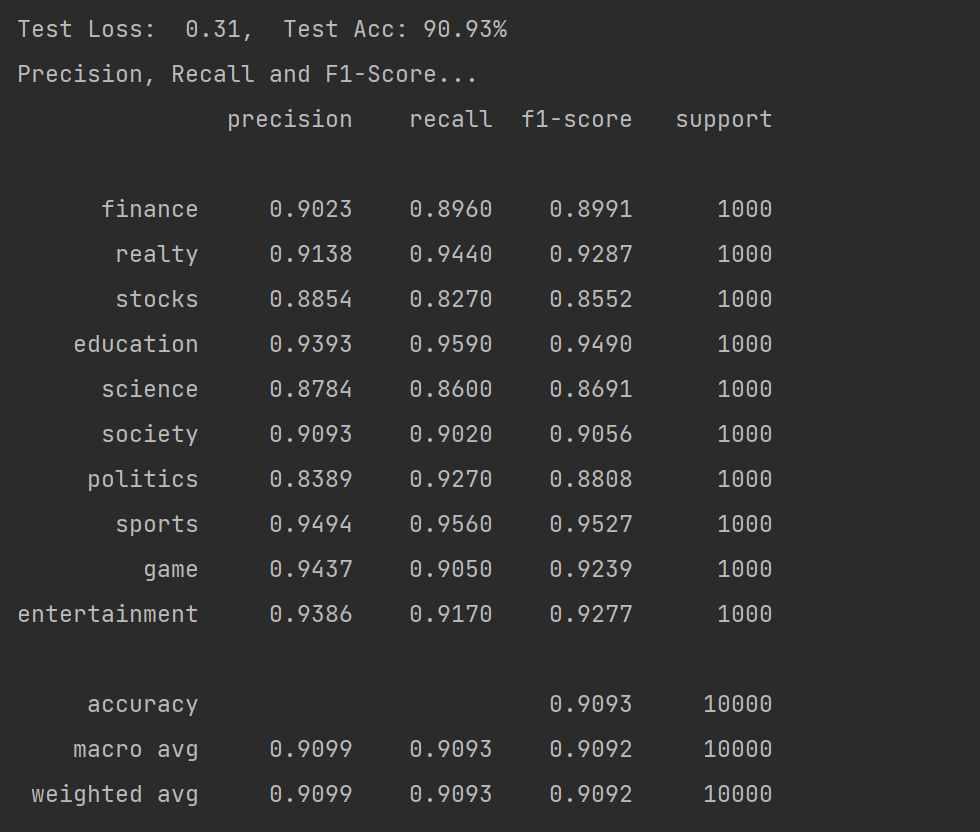
**图3-1 CNN模型结构图**

### 2、CNN模型代码

|  |
| --- |
| class Model(nn.Module):     def \_\_init\_\_(self, config):         super(Model, self).\_\_init\_\_()         if config.embedding\_pretrained is not None:             self.embedding = nn.Embedding.from\_pretrained(config.embedding\_pretrained, freeze=False)         else:             self.embedding = nn.Embedding(config.n\_vocab, config.embed, padding\_idx=config.n\_vocab - 1)         self.convs = nn.ModuleList(             [nn.Conv2d(1, config.num\_filters, (k, config.embed)) for k in config.filter\_sizes])         self.dropout = nn.Dropout(config.dropout)         self.fc = nn.Linear(config.num\_filters \* len(config.filter\_sizes), config.num\_classes)      def conv\_and\_pool(self, x, conv):         x = F.relu(conv(x)).squeeze(3)         x = F.max\_pool1d(x, x.size(2)).squeeze(2)         return x      def forward(self, x):         out = self.embedding(x[0])         out = out.unsqueeze(1)         out = torch.cat([self.conv\_and\_pool(out, conv) for conv in self.convs], 1)          out = self.dropout(out)         out = self.fc(out)         return out |

### 3、CNN模型下文本分类结果

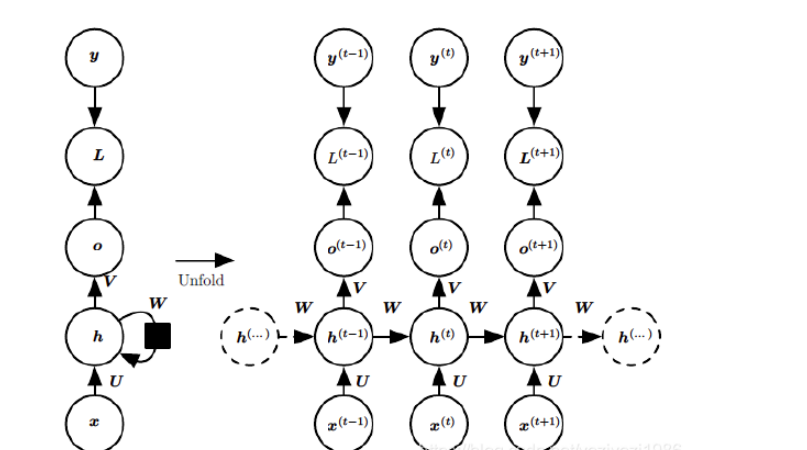
下图为测试集的实验结果，包含各个类别的精确率、召回率、F1值、准确率等衡量指标。各个类别数据量均为1000，其中sports类别分类效果最好。测试集总体准确率为90.93%。



**图3-2 CNN模型实验结果评价**

## RNN模型结构及文本分类的实验结果

### 1、RNN模型结构



**图3-3 RNN模型结构图**

其中:

1) Xt代表序列t时刻训练样本的输入；

2) ht代表序列t时刻模型的隐藏状态；

3) ot代表序列t时刻模型的输出；

4) Lt代表序列t时刻模型的损失函数，模型整体的损失函数是所有Lt相加所得；

5）yt代表序列t时刻模型的真实输出；

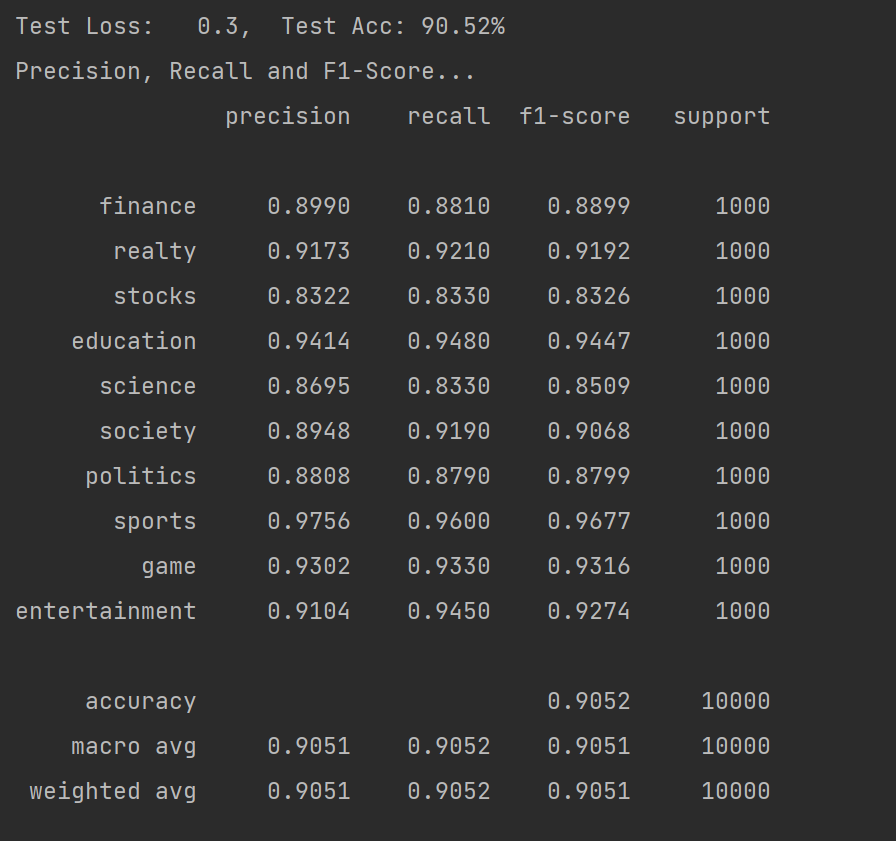
6）U,W,V是模型的参数，它们在整个RNN网络中是共享的，也正是因为它们共享，体现了RNN模型的“循环反馈”思想。

### 2、RNN模型代码

|  |
| --- |
| def \_\_init\_\_(self, config):         super(Model, self).\_\_init\_\_()         if config.embedding\_pretrained is not None:             self.embedding = nn.Embedding.from\_pretrained(config.embedding\_pretrained, freeze=False)         else:             self.ebedding = nn.Embedding(config.n\_vocab, config.embed, padding\_idx=config.n\_vocab - 1)         self.lstm = nn.LSTM(config.embed, config.hidden\_size, config.num\_layers,                             bidirectional=True, batch\_first=True, dropout=config.dropout)         self.fc = nn.Linear(config.hidden\_size \* 2, config.num\_classes)      def forward(self, x):         x, \_ = x         out = self.embedding(x)  # [batch\_size, seq\_len, embeding]=[128, 32, 300]         out, \_ = self.lstm(out)         out = self.fc(out[:, -1, :])  # 句子最后时刻的 hidden state         return out |

### 3、RNN模型下文本分类结果

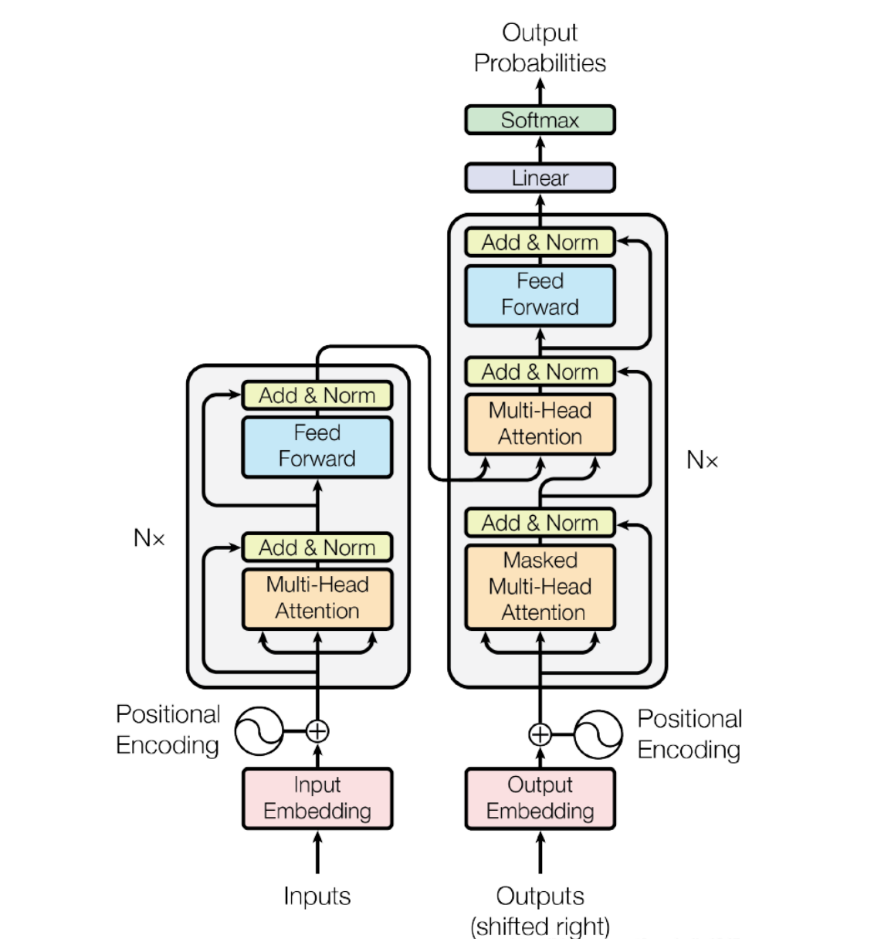
下图为测试集的实验结果，包含各个类别的精确率、召回率、F1值、准确率等衡量指标。各个类别数据量均为1000，其中sports类别分类效果最好。测试集总体准确率为90.52%。



**图3-4 RNN实验结果评价**

## Transformer模型结构及文本分类的实验结果

### 1、Transformer模型结构



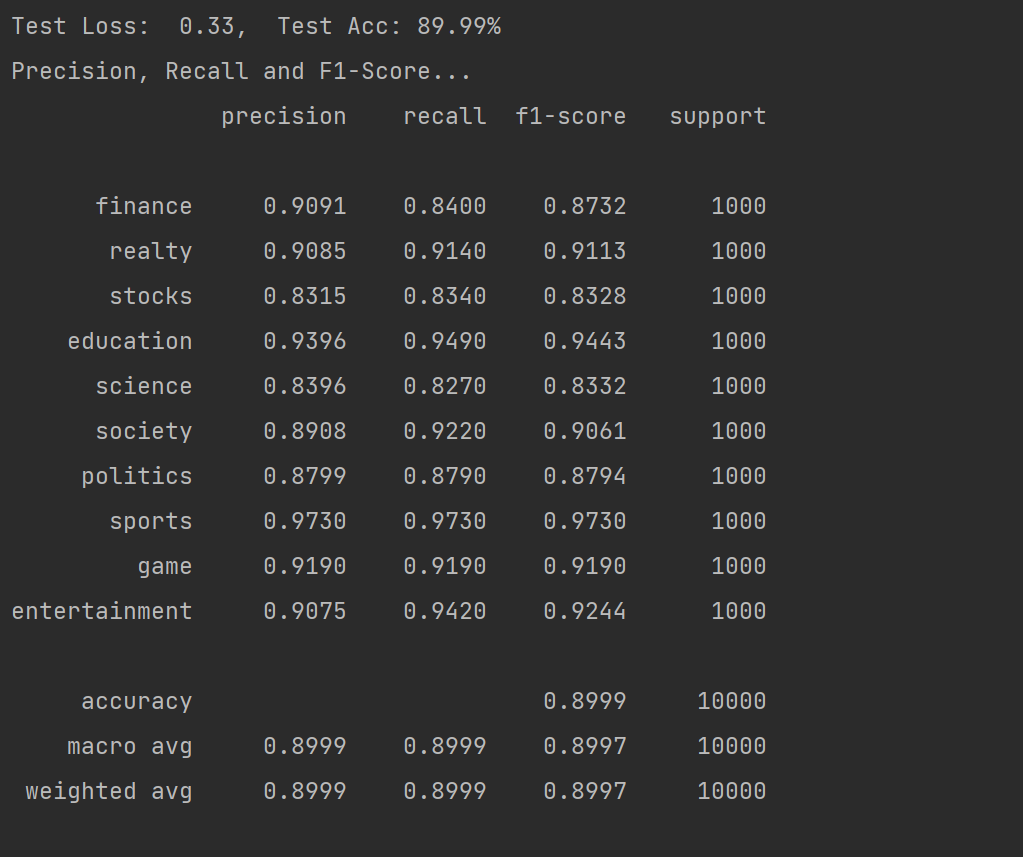
**图3-5 RNN模型结构图**

### 2、Transformer模型代码

|  |
| --- |
| def \_\_init\_\_(self, config):         super(Model, self).\_\_init\_\_()         if config.embedding\_pretrained is not None:             self.embedding = nn.Embedding.from\_pretrained(config.embedding\_pretrained, freeze=False)         else:             self.embedding = nn.Embedding(config.n\_vocab, config.embed, padding\_idx=config.n\_vocab - 1)    self.postion\_embedding = Positional\_Encoding(config.embed, config.pad\_size, config.dropout, config.device)         self.encoder = Encoder(config.dim\_model, config.num\_head, config.hidden, config.dropout)         self.encoders = nn.ModuleList([epcopy(self.encoder)             # Encoder(config.dim\_model, config.num\_head, config.hidden, config.dropout)             for \_ in range(config.num\_encoder)])          self.fc1 = nn.Linear(config.pad\_size \* config.dim\_model, config.num\_classes)         # self.fc2 = nn.Linear(config.last\_hidden, config.num\_classes)         # self.fc1 = nn.Linear(config.dim\_model, config.num\_classes)      def forward(self, x):         out = self.embedding(x[0])         out = self.postion\_embedding(out)         for encoder in self.encoders:             out = encoder(out)         out = out.view(out.size(0), -1)         # out = torch.mean(out, 1)         out = self.fc1(out)         return out |

### 3、Transformer模型下文本分类结果

下图为测试集的实验结果，包含各个类别的精确率、召回率、F1值、准确率等衡量指标。各个类别数据量均为1000，其中sports类别分类效果最好。测试集总体准确率为89.99%。



**图3-6** **Transformer实验结果评价**

## 总结与分析

从上面实验结果可以看出，在本数据集的条件下，在三个模型中，CNN模型（90.93%）测试集准确率较RNN（90.52%），Transformer（89.99%）两个模型分类效果高。CNN对处理高维数据无压力，在本实验中解决捕获更远距离的特征，从而确保了良好的性能。RNN和Transformer稍显不足，但差距并不大。

三个模型对于分类的各个类别的准确率高低趋于一致，总体上相近，在本实验中对sports类别分类效果都是最高的，其中虽然Transformer总体准确率较低，但针对sports该类别实验效果最佳，这也显示出Transformer对语义特征提取能力，综合特征抽取能力，对语法、句法的理解能力显著超过另外两种。