本实验用到了工具库keras和sklearn，就CNN对文本的内容和情感的分类做对比试验

ContentClassification

1只用CNN

模型结构：

model = Sequential()

model.add(Embedding(len(word\_index)+1,EMBEDDING\_DIM,input\_length=MAX\_SEQUENCE\_LENGTH))

model.add(Dropout(0.2))

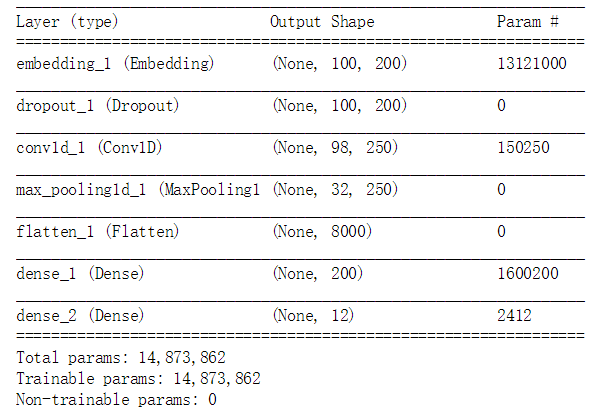
model.add(Conv1D(250, 3, padding='valid', activation='relu', strides=1))

model.add(MaxPooling1D(3))

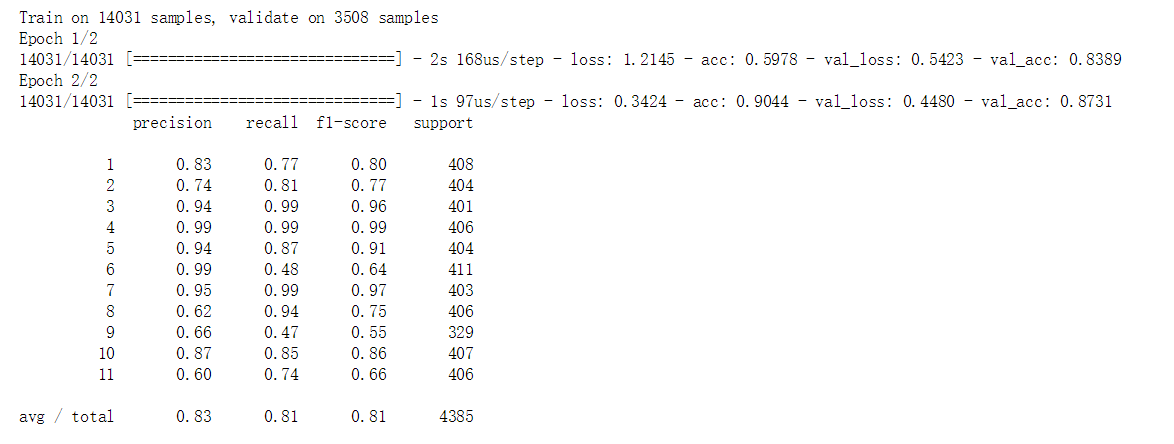
model.add(Flatten())

model.add(Dense(EMBEDDING\_DIM, activation='relu'))

model.add(Dense(labels.shape[1], activation='softmax'))



性能结果:



2 cnn+fasttext:

模型结构：

model = Sequential()

model.add(embedding\_layer)

model.add(Dropout(0.2))

model.add(Conv1D(250, 3, padding='valid', activation='relu', strides=1))

model.add(MaxPooling1D(3))

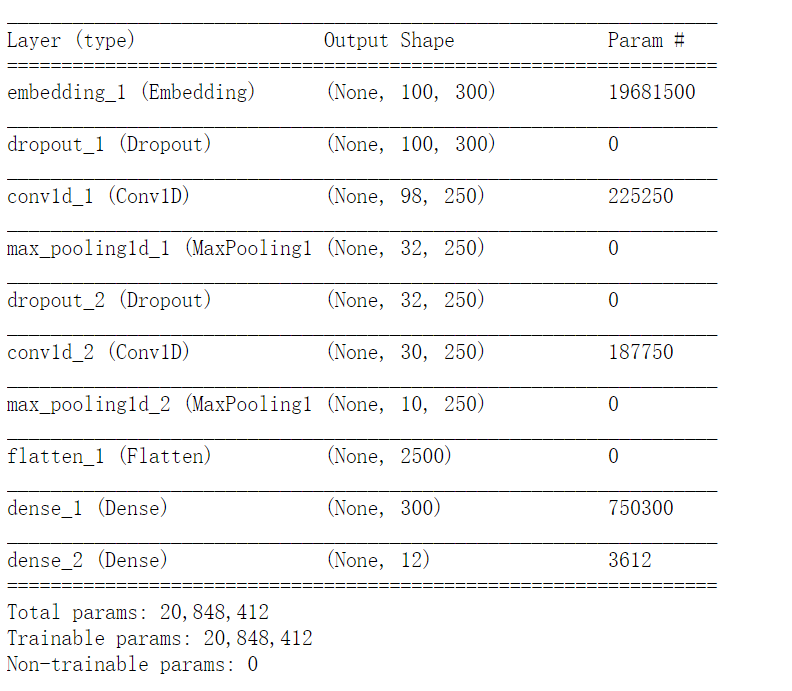
model.add(Dropout(0.2))

model.add(Conv1D(250, 3, padding='valid', activation='relu', strides=1))

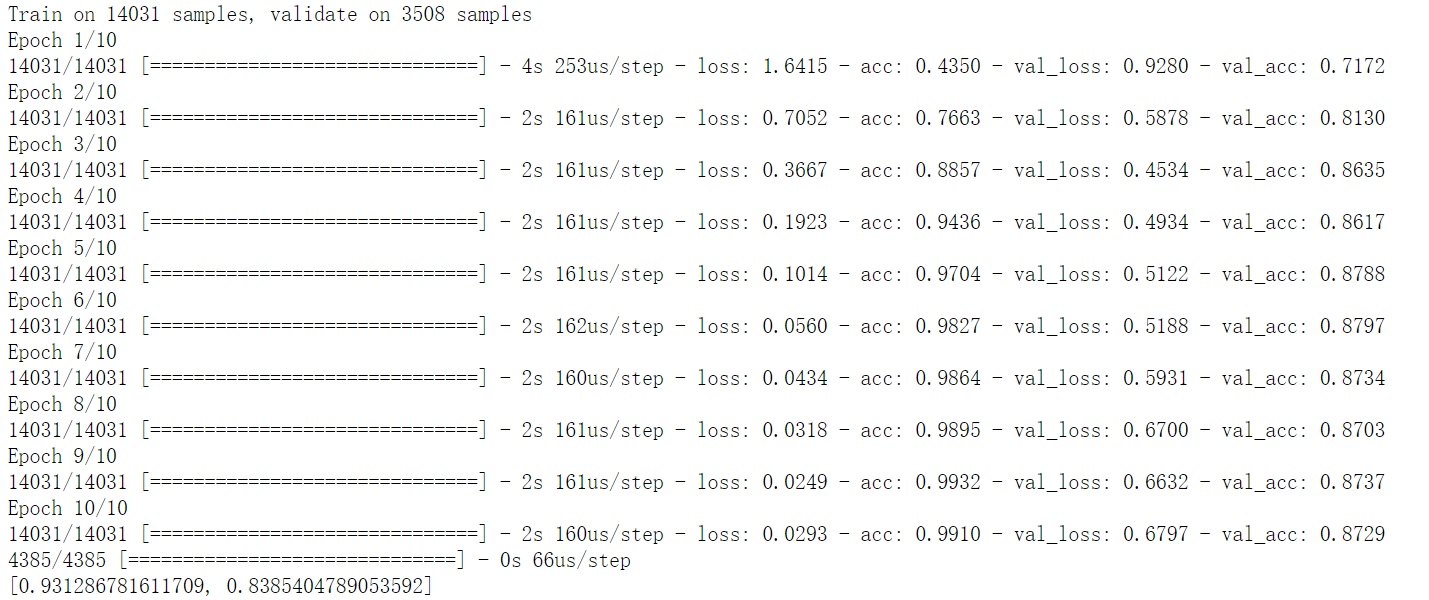
model.add(MaxPooling1D(3))

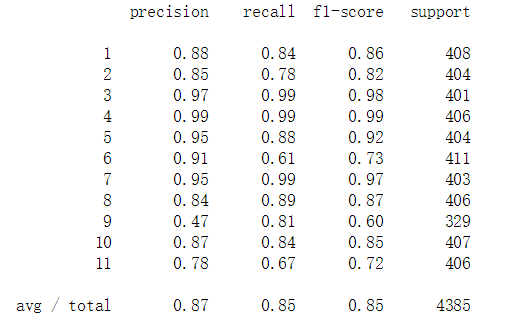
model.add(Flatten())

model.add(Dense(EMBEDDING\_DIM, activation='relu'))

model.add(Dense(labels.shape[1],activation='softmax'))

性能结果：





只用cnn和将embedding层参数减少替换成fasttext词向量模型的cnn相比：

后者从精确率，召回率以及F1\_score上的表现都更好

EmotionClassification

cnn\_only:

模型结构：

model = Sequential()

model.add(Embedding(len(word\_index)+1,EMBEDDING\_DIM,input\_length=MAX\_SEQUENCE\_LENGTH))

model.add(Dropout(0.2))

model.add(Conv1D(250, 3, padding='valid', activation='relu', strides=1))

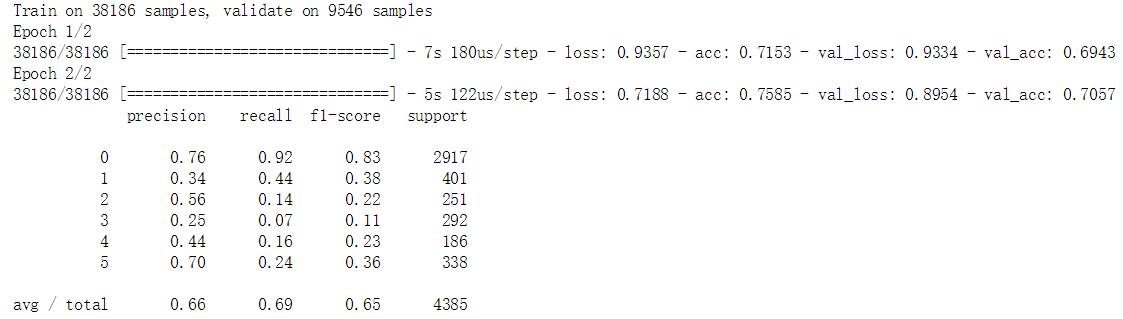
model.add(MaxPooling1D(3))

model.add(Flatten())

model.add(Dense(EMBEDDING\_DIM, activation='relu'))

model.add(Dense(labels.shape[1], activation='softmax'))

性能结果：



fasttext+cnn:

模型结构：

model = Sequential()

model.add(embedding\_layer)

model.add(Dropout(0.2))

model.add(Conv1D(250, 3, padding='valid', activation='relu', strides=1))

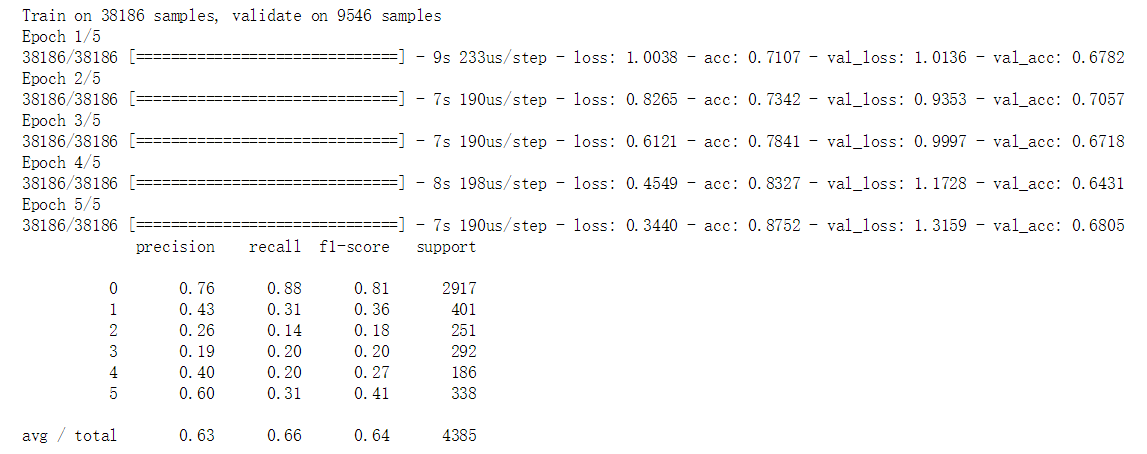
model.add(MaxPooling1D(3))

model.add(Flatten())

model.add(Dense(EMBEDDING\_DIM, activation='relu'))

model.add(Dense(labels.shape[1], activation='softmax'))

性能结果：



只用cnn和将embedding层参数减少替换成fasttext词向量模型的cnn相比：

前者从精确率，召回率以及F1\_score上的表现都更好。这说明在文本的情感分类上，fasttext并不能起到比较好的优化效果。