基于中文文本的表情预测*

王明[1+]

[1](南京大学 计算机科学与技术系,南京 210046)

Emoji Prediction based on Chinese Text*

WANG Ming[1+]

[1](Department of Computer Science and Technology, Nanjing University, Nanjing 210046, China)

+ Corresponding author: Phn: +86-13851085654, E-mail: 1090616897@qq.com, http://www.nju.edu.cn

Abstract: Characters representing feelings like Emoji have been an indispensable part of modern human's social life. Character not only can assist us to express richer feelings, but also can make the expression more vivid. However, with the increasing creation of Emojis, sometimes we don't know how to choose one which can properly show our sentiments. So it is meaningful to design a technique to predict emoji for a specific Chinese text. By Chinese text partitioning, text feature extracting, word vector training, classification learning like multi-layer perceptron, we can predict one emoji which match one specific Chinese text to some extent.

Key words: emoji prediction; Chinese text; text feature extraction; classification learning; multi-layer perceptron

摘 要: 表示情感的符号比如 Emoji 已经成了现代人类社会交际中必不可少的一部分，表情符号的使用不仅可以帮助我们表达更为丰富的内容，并且可以使这种情绪表达得更加生动形象。但是随着表情符号越来越多地被创造出来，有时候我们不知道该用什么表情，不知道什么表情可以精确地表达我们现在的心情。因此做基于中文文本的表情预测是很有意义的，通过中文分词，文本特征提取，训练出体现文本联系的词向量，然后使用传统数据挖掘分类学习的方法如多层感知机，便可以一定程度地预测出与一段中文文本相匹配的表情。

关键词: 表情预测;中文文本;文本特征提取;分类学习;多层感知机

1 挖掘任务理解

我们有标记好的训练集，包含文本和对应的表情，目标是去预测测试集中每一行文本多对应的表情。这显然是一个分类任务，大致的过程应该是：对每一行文本进行特征提取，表达成向量的形式，然后使用数据挖掘的方法去训练生成模型，最后使用此模型去预测测试集的表情。（过程见图 1）

我认为该任务的难点有两个，一个是文本特征的提取，这一步可以说很大程度决定了最终结果的好坏。这一步必须要满足一下两点：第一，提取出的特征必须能很好地反映出两段文本之间的关联性的强弱。第二，提取出的特征必须能表达成一维向量的形式，因为要满足分类算法的输入形式。第二个是分类算法的设计，必须考虑使用什么样的分类器以及过拟合等问题。

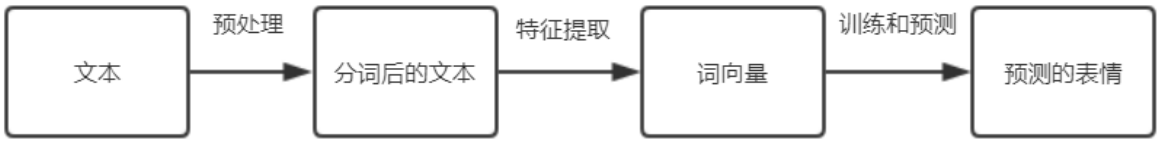

图1 基本挖掘过程

2 文本特征提取

2.1 数据预处理

从横向来看，对于一段中文文本最简单直接的方法就是保留所有中文，去除其他一切非中文的符号（包括英文符号、标点符号、特殊符号等等）。但是这样做显然会损失一些信息，因为像‘!’‘？’‘。。。。。’等符号带有强烈的感情色彩，如果去掉，必然会使得结果失准。因此我选择在保留中文的基础上，保留一些带有强烈感情色彩的、有意义的标点符号。

从纵向来看，只要稍微翻阅一下训练集的文本，就会发现其实训练集中的某些分类本身就是不准确的，比如一些明明表达的是哭的含义，却匹配了一个笑脸，而有些表达的是开心的含义，却匹配了一个悲伤的表情，还有一些很莫名其妙的一串符号也匹配了一个没看懂的表情。总结起来就是训练集的噪声是很大的，有很多脏数据，但是棘手的是这些数据还没有表达成向量的形式，因此无法通过一系列的处理噪声和离群点分析的方法来去除这些噪声。另一种想法是随机使用部分的训练集，这样存在两个问题：第一个是随机选取出来的有几率不覆盖所有的分类，对于有些算法来讲，没有覆盖的类永远都不会被预测；第二个是由于噪声的分布未知，因此随机选取并不一定能降低噪声的比例。经过思考之后还是决定使用全部的训练集来进行训练，但是在后续使用强分类器的时候一定要注意过拟合的问题，因为如果过拟合，就会放大了噪声的影响，从而影响预测的精准度。

2.2 词向量生成

在生成词向量这一块，我使用的是 gensim 下的 word2vec 库，该库的作用是将词转换成词向量，并且这种转换可以自己设定维数，其算法需要做的是根据词语所在的上下文，将词语从高维映射到低维，并且使得在低维上能够保持高维的关系，该算法的输入是独热编码的超高维向量（对于每个词语，只有一位为 1，其余都是 0），然后结合词语所在的上下文（可以通过半径参数来控制上下文的范围），使用带有隐层的神经网络算法来进行训练，最终得到了带有语义的维数固定且可控的词向量。

要使用这个词向量模型进行训练的话，我们需要首先对一个中文句子进行分词，使用的是 python 很优秀的中文分词库 jieba，输入是一段中文文本，输出是分好的词组成的列表。使用 word2vec 进行训练便得到了每个词的词向量。

那么现在一句话每个词的词向量都有了，那么该怎么得到这句话的词向量呢，最常用的就是赋予每个词同样的权重，取他们的向量的求和平均，以这个平均的结果作为这句话的向量。

2.3 其他向量生成方法

文本特征提取的方法出了上面叙述的先分词，再训练得到每个词的词向量，然后求和平均得到每句话的句向量，还有一种处理长文本的方法，叫做 doc2vec，它的基本原理和 word2vec 十分相似，只不过不是利用一个词和它上下文的关系训练生成这个词的向量，而是直接利用句间关系生成句向量。

我尝试使用了 doc2vec 后发现效果很差，原因可能是使用了同一个表情的两个句子的句向量可能会相差很大。比如“我爱你”和“今天我收到了很漂亮的礼物，是一条施华洛世奇的手链，我真的太喜欢了。”这两句话都使用了“心”这个表情，但是 doc2vec 训练的时候就大概率会相差很大，因为从字数和文本本身的组

成方面就相差很大。

2.4 实现细节

数据预处理时使用了正则表达式，对一段文本匹配需要去除的字符，然后将其去掉就得到了需要保留的字符。当然在处理之前需要将文本数据解码成 UTF-8-SIG 格式。然后使用 jieba 库进行分词，得到分词之后，放入 word2vec 模型进行训练得到每个词的词向量。最后对于每句话所有词的词向量进行求和平均，得到代表这句话的向量。（过程见图 2）

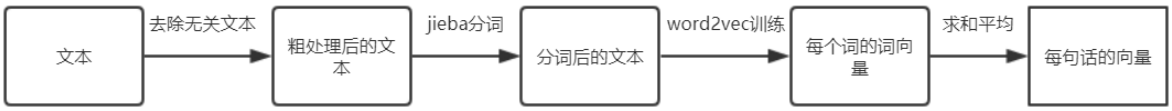

图2 文本特征提取过程

3 分类算法

3.1 MLP

现在已经完成了中文文本的特征提取，并且处理成了向量的形式，那么接下来要做的就是构建向量和表情之间的映射了，这样就完全成了一个标准的分类问题了。为了提高预测的效果，这里使用强分类器，而最容易想到的就是神经网络算法。之前已经探讨过，由于训练集的噪声数据不能忽略，因此不能过拟合，因为过拟合之后势必会将结果一定程度偏向噪声数据，从而降低效果。

我使用的是 sklearn.neural_network 库下的 MLPClassifier 分类器，也就是多层感知机神经网络，通过不断重复迭代输入，不断调整连接权重，直到结果收敛。（使用损失函数和代价函数来衡量训练是否完成）在参数方面，主要需要控制的是隐层的层数和每层的节点个数。事实证明，层数只需 2 到 3 层，层数越大，不仅训练时间长，而且效果反而不好，同时每层的节点数也不应该过多。

模型训练完成之后，就可以进行预测了。对于需要预测的每一句文本，也和训练集一样进行预处理、中文分词、转换成向量，然后将这个向量输入到模型，得到的便是预测的表情。

3.2 其他分类算法

MLP 神经网络算法是试过的众多的方法中 F1-Mean Score 最高的算法。我试过的其他分类方法包括决策树算法、朴素贝叶斯算法、随机森林算法、自己手写的聚类+KNN 的算法，这些非神经网络效果都不是很好。其中效果最好的是随机森林算法，可以看出集成学习还是很厉害的。

3.3 实现细节

使用 MLP 算法比较简单，直接调用 sklearn.neural_network 库下的 MLPClassifier 分类器，设置层数和每层的节点数，将之前训练得到的训练集的词向量和对应的表情输入用 fit 函数进行模型的训练，然后用 predict 函数对测试集的词向量进行预测。

4 实验结果

4.1 最终结果

最终得分最高的是使用的 MLP 算法，参数为 hidden_layer_sizes = (115,115)训练出来的模型，在 Kaggle 网站上，Public Score 是 0.17060，Private Score 是 0.17062。

4.2 各算法的效果对比

衡量标准采用 Kaggle 网站上的 F1-Score

1、多层感知机算法（MLP）的比较：

文本特征提取	分类器模型参数	Private Score
无预处理 + word2vec	hidden_layer_sizes = (100,)	0.15503
无预处理 + word2vec	hidden_layer_sizes = (100,100,100,100,100,100,100,100)	0.15546
无预处理 + word2vec	hidden_layer_sizes = (100,100)	0.15721
无预处理 + doc2vec	hidden_layer_sizes = (100,)	0.13790
只保留中文 + word2vec	hidden_layer_sizes = (100,100)	0.16265
保留有意义字符 + word2vec	hidden_layer_sizes = (115,115)	0.17062

表 1　MLP 的比较

2、随机森林算法（RandomForest）的比较：

文本特征提取	分类器模型参数	Private Score
无预处理 + word2vec	默认参数	0.08494
无预处理 + word2vec	n_estimators = 30, max_depth = 20	0.13278
无预处理 + word2vec	n_estimators = 50, max_depth = 30	0.12928
无预处理 + word2vec	n_estimators = 50, max_depth = 15	0.14342
无预处理 + doc2vec	n_estimators = 30, max_depth = 20	0.10291
无预处理 + doc2vec	n_estimators = 50, max_depth = 20	0.10945

表 2　RandomForest 的比较

3、各种算法的效果对比

分类算法	分类器模型参数	Private Score
MLP	hidden_layer_sizes = (115,115)	0.17062
RandomForest	n_estimators = 50, max_depth = 15	0.14342
Naïve Bayes	no parameters	0.04735
DecisionTree	no parameters	0.05276
self-made cluster + kNN	no parameters	0.06074

表 3　各种算法的比较

4.3 结论

1、 文本特征提取的差别可以很大程度地影响最终的预测效果（比如对初始的文本做去除特殊符号的预处理和不做处理结果会相差很大）。

2、 生成每句话的向量的时候，采用 word2vec 的求和平均比 doc2vec 的效果要好很多。

3、 强分类器的效果通常比弱分类器好，神经网络算法的效果通常比非神经网络算法好，在非神经网络算法中，集成学习（如随机森林）的效果通常会更好。

4、 使用神经网络算法的时候并不是层数越多越好，因为训练数据本身脏数据很多，越是精确拟合越可能加大噪声的比重。

5、 这次挖掘任务真切的感受到有一章 PPT 讲 Data Preprocessing 里有一句话：“Data precprocessing usually occupies about 70% workload in a data mining task”。真切感受到数据的重要性，准确无误的数据比用什么算法来的更加重要。感觉大多数同学和我一样都把精力放在寻找更好的分类算法上，而忽略了数据的重要性。数据的质量基本决定了预测效果所能达到的上限。

6、 我认为要进一步提高预测效果，需要思考：（1）对于一句话，如何赋予不同的词已不同的权重（例如对于强烈表达情绪的词赋予更高权重）来求这句话的句向量。（2）对于一句话，能否对于中文、标点符号、英文进行分别的处理使得分析和比较更加合理。（3）对于训练集进行纵向处理，去除那些很有可能成为噪声的项。（例如对属于同一个类的句子的句向量进行聚类和离群点分析，剔除与聚类中心相距较远的点）

5 结束语

本文以一个分类问题的视角，使用 jieba 中文分词、word2vec 词向量模型完成中文文本到向量的转换，使用 Multi-Layer Perceptron 神经网络算法进行训练和预测，从而完成了对中文文本的表情预测。

致谢 非常感谢挖掘任务期间助教各方面的贴心的解答和帮助。

References:

附中文参考文献:

[1] 缺省之名，通俗理解 word2vec，https://www.jianshu.com/p/471d9bfbd72f

[2] Dominic221，神经网络基础-多层感知器(MLP)，https://blog.csdn.net/weixin_38206214/article/details/81137911