

## 02 | 无需任何机器学习，如何利用大语言模型做情感分析？

2023-03-22 徐文浩 来自北京

《AI大模型之美》



你好，我是徐文浩。

上一讲我们看到了，大型语言模型的接口其实非常简单。像 OpenAI 就只提供了 Complete 和 Embedding 两个接口，其中，Complete 可以让模型根据你的输入进行自动续写，Embedding 可以将你输入的文本转化成向量。

不过到这里，你的疑问可能就来了。不是说现在的大语言模型很厉害吗？传统的自然语言处理问题都可以通过大模型解决。可是用这么简单的两个 API，能够完成原来需要通过各种 NLP 技术解决的问题吗？比如情感分析、文本分类、文章聚类、摘要撰写、搜索，这一系列问题怎么通过这两个接口解决呢？

别急，在接下来的几讲里，我会告诉你，怎么利用大语言模型提供的这两个简单的 API 来解决传统的自然语言处理问题。这一讲我们就先从一个最常见的自然语言处理问题——“情感分析”开始，来看看我们怎么把大语言模型用起来。

## 传统的二分类方法：朴素贝叶斯与逻辑回归

“情感分析”问题，是指我们根据一段文字，去判断它的态度是正面的还是负面的。在传统的互联网产品里，经常会被用来分析用户对产品、服务的评价。比如大众点评里面，你对餐馆的评论，在京东买个东西，你对商品的评论，都会被平台拿去分析，给商家或者餐馆的评分做参考。也有些品牌，会专门抓取社交网络里用户对自己产品的评价，来进行情感分析，判断消费者对自己的产品评价是正面还是负面的，并且会根据这些评价来改进自己的产品。

对于“情感分析”类型的问题，传统的解决方案就是**把它当成是一个分类问题**，也就是先拿一部分评论数据，人工标注一下这些评论是正面还是负面的。如果有个用户说“这家餐馆真好吃”，那么就标注成“正面情感”。如果有个用户说“这个手机质量不好”，那么就把对应的评论标注成负面的。

我们把标注好的数据，喂给一个机器学习模型，训练出一组参数。然后把剩下的没有人工标注过的数据也拿给训练好的模型计算一下。模型就会给你一个分数或者概率，告诉你这一段评论的感情是正面的，还是负面的。

可以用来做情感分析的模型有很多，这些算法背后都是基于某一个数学模型。比如，很多教科书里，就会教你用**朴素贝叶斯算法**来进行垃圾邮件分类。朴素贝叶斯的模型，就是简单地统计每个单词和好评差评之间的条件概率。一般来说，如果一个词语在差评里出现的概率比好评里高得多，那这个词语所在的评论，就更有可能是一个差评。

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$

$$P(c|X) \propto P(x_1|c) \times P(x_2|c) \times \cdots \times P(x_n|c) \times P(c)$$

假设我们有一个训练集包含 4 封邮件，其中 2 封是垃圾邮件，2 封是非垃圾邮件。训练集里的邮件包含这些单词。

序号	邮件类型	出现的单词
邮件1	垃圾邮件	buy, money, offer, secret
邮件2	垃圾邮件	buy, secret, sell, money
邮件3	普通邮件	offer, book, sell
邮件4	普通邮件	book, study, exam

然后来了一封新邮件，里面的单词是：buy、money、sell。

通过这些单词出现的概率，我们很容易就可以预先算出这封邮件是垃圾邮件还是普通邮件。

$$P(\text{buy}|\text{垃圾}) = 2 \div 2 = 1$$

$$P(\text{money}|\text{垃圾}) = 2 \div 2 = 1$$

$$P(\text{sell}|\text{垃圾}) = 1 \div 2 = 0.5$$

$$P(\text{buy}|\text{普通}) = 0 \div 2 = 0$$

$$P(\text{money}|\text{普通}) = 0 \div 2 = 0$$

$$P(\text{sell}|\text{普通}) = 1 \div 2 = 0.5$$

然后我们把这封邮件里所有词语的条件概率用全概率公式乘起来，就得到了这封邮件是垃圾邮件还有普通邮件的概率。

$$P(\text{垃圾}|X) \propto P(\text{buy}|\text{垃圾}) \times P(\text{money}|\text{垃圾}) \times P(\text{sell}|\text{垃圾}) \times P(\text{垃圾}) = 1$$

$$P(\text{普通}|X) \propto P(\text{buy}|\text{普通}) \times P(\text{money}|\text{普通}) \times P(\text{sell}|\text{普通}) \times P(\text{普通}) = 0$$

在这里，我们发现  $P(\text{垃圾}|X) > P(\text{普通}|X)$ ，而且  $P(\text{普通}|X)$  其实等于 0。那如果用朴素贝叶斯算法，我们就会认为这封邮件 100% 是垃圾邮件。如果你觉得自己数学不太好，这个例子没有看明白也没有关系，因为我们接下来的 AI 应用开发都不需要预先掌握这些数学知识，所以不要有心理负担。

类似的，像逻辑回归、随机森林等机器学习算法都可以拿来做分类。你在网上，特别是 Kaggle 这个机器学习比赛的网站里，可以搜索到很多其他人使用这些传统方法来设计情感分析的解决方案。这些方案都以 Jupyter Notebook 的形式出现，我在这里放个 [🔗 链接](#)，你有兴趣的话也可以去研究一下。

## 传统方法的挑战：特征工程与模型调参

但这些传统的机器学习算法，想要取得好的效果，还是颇有门槛的。除了要知道有哪些算法可以用，还有两方面的工作非常依赖经验。

### 特征工程

第一个是特征工程。对于很多自然语言问题，如果我们只是拿一段话里面是否出现了特定的词语来计算概率，不一定是最合适的。比如“这家餐馆太糟糕了，一点都不好吃”和“这家餐馆太好吃了，一点都不糟糕”这样两句话，从意思是完全相反的。但是里面出现的词语其实是相同的。在传统的自然语言处理中，我们会通过一些特征工程的方法来解决这个问题。

比如，我们不只是采用单个词语出现的概率，还增加前后两个或者三个相连词语的组合，也就是通过所谓的 2-Gram (Bigram 双字节词组) 和 3-Gram (Trigram 三字节词组) 也来计算概率。在上面这个例子里，第一句差评，就会有“太”和“糟糕”组合在一起的“太糟糕”，以及“不”和“好吃”组合在一起的“不好吃”。而后面一句里就有“太好吃”和“不糟糕”两个组合。有了这样的 2-Gram 的组合，我们判断用户好评差评的判断能力就比光用单个词语是否出现要好多了。

这样的特征工程的方式有很多，比如**去除停用词**，也就是“的地得”这样的词语，**去掉过于低频的词语**，比如一些偶尔出现的专有名词。或者对于有些词语特征采用 **TF-IDF** (词频 - 逆文档频率) 这样的统计特征，还有在英语里面对不同态的单词统一换成现在时。

不同的特征工程方式，在不同的问题上效果不一样，比如我们做情感分析，可能就需要保留标点符号，因为像“！”这样的符号往往蕴含着强烈的情感特征。但是，这些种种细微的技巧，让我们在想要解决一个简单的情感分析问题，也需要撰写大量文本处理的代码，还要了解针对当前特定场景的技巧，这非常依赖工程师的经验。

## 机器学习相关经验

第二个就是你需要有相对丰富的机器学习经验。除了通过特征工程设计更多的特征之外，我们还需要了解很多机器学习领域里常用的知识和技巧。比如，我们需要将数据集切分成训练（Training）、验证（Validation）、测试（Test）三组数据，然后通过 AUC 或者混淆矩阵（Confusion Matrix）来衡量效果。如果数据量不够多，为了训练效果的稳定性，可能需要采用 K-Fold 的方式来进行训练。

如果你没有接触过机器学习，看到这里，可能已经看懵了。没关系，上面的大部分知识你未来可能都不需要了解了，因为我们有了大语言模型，可以通过它提供的 Completion 和 Embedding 这两个 API，用不到 10 行代码就能完成情感分析，并且能获得非常好的效果。

## 大语言模型：20 行代码的情感分析解决方案


通过大语言模型来进行情感分析，最简单的方式就是利用它提供的 Embedding 这个 API。这个 API 可以把任何你指定的一段文本，变成一个大语言模型下的向量，也就是用一组固定长度的参数来代表任何一段文本。

我们需要提前计算“好评”和“差评”这两个字的 Embedding。而对于任何一段文本评论，我们也都可以通过 API 拿到它的 Embedding。那么，我们把这段文本的 Embedding 和“好评”以及“差评”通过余弦距离（Cosine Similarity）计算出它的相似度。然后我们拿这个 Embedding 和“好评”之间的相似度，去减去和“差评”之间的相似度，就会得到一个分数。如果这个分数大于 0，那么说明我们的评论和“好评”的距离更近，我们就可以判断它为好评。如果这个分数小于 0，那么就是离差评更近，我们就可以判断它为差评。

下面我们就用这个方法分析一下两条在京东上购买了 iPhone 用户的评论。



这个使用大模型的方法一共有 20 行代码，我们看看它能否帮助我们快速对这两条评论进行情感分析。

 复制代码

```
1 import openai
2 import os
3 from openai.embeddings_utils import cosine_similarity, get_embedding
4
5 # 获取访问open ai的密钥
6 openai.api_key = os.getenv("OPENAI_API_KEY")
7 # 选择使用最小的ada模型
8 EMBEDDING_MODEL = "text-embedding-ada-002"
9
10 # 获取"好评"和"差评"的
11 positive_review = get_embedding("好评")
12 negative_review = get_embedding("差评")
13
14 positive_example = get_embedding("买的银色版真的很好看，一天就到了，晚上就开始拿起来完系统")
15 negative_example = get_embedding("降价厉害，保价不合理，不推荐")
16
17 def get_score(sample_embedding):
18     return cosine_similarity(sample_embedding, positive_review) - cosine_similarity
19
20 positive_score = get_score(positive_example)
21 negative_score = get_score(negative_example)
22
23 print("好评例子的评分 : %f" % (positive_score))
24 print("差评例子的评分 : %f" % (negative_score))
```

输出结果：

[复制代码](#)

```
1 好评例子的评分 : 0.070963
2 差评例子的评分 : -0.081472
```

正如我们所料，京东上的好评通过 Embedding 相似度计算得到的分数是大于 0 的，京东上面的差评，这个分数是小于 0 的。

这样的方法，是不是特别简单？我们再拿刚才的例子试一下，看看这个方法是不是对所有词语都管用，只是出现的位置不同但含义截然相反的评论，能得到什么样的结果。

[复制代码](#)

```
1 good_restraurant = get_embedding("这家餐馆太好吃了，一点都不糟糕")
2 bad_restraurant = get_embedding("这家餐馆太糟糕了，一点都不好吃")
3
4 good_score = get_score(good_restraurant)
5 bad_score = get_score(bad_restraurant)
6 print("好评餐馆的评分 : %f" % (good_score))
7 print("差评餐馆的评分 : %f" % (bad_score))
```

输出结果：

[复制代码](#)

```
1 好评餐馆的评分 : 0.062719
2 差评餐馆的评分 : -0.074591
```

可以看到，虽然两句话分别是“太好吃”“不糟糕”和“太糟糕”“不好吃”，其实词语都一样，但是大语言模型一样能够帮助我们判断出来他们的含义是不同的，一个更接近好评，一个更接近差评。


## 更大的数据集上的真实案例

在这里，我们只举了几个例子，看起来效果还不错。这会不会只是我们运气好呢？我们再来拿一个真实的数据集验证一下，利用这种方法进行情感分析的准确率能够到多少。



下面这段代码，是来自 OpenAI Cookbook 里面的一个例子。它是用同样的方法，来判断亚马逊提供的用户对一些食物的评价，这个评价数据里面，不只有用户给出的评论内容，还有用户给这些食物打了几颗星。这些几颗星的信息，正好可以拿来验证我们这个方法有多准。对于用户打出 1~2 星的，我们认为是差评，对于 4~5 星的，我们认为是好评。

我们可以通过 Pandas，将这个 CSV 数据读取到内存里面。为了避免重新调用 OpenAI 的 API 浪费钱，这个数据集里，已经将获取到的 Embedding 信息保存下来了，不需要再重新计算。

 复制代码

```
1 import pandas as pd
2 import numpy as np
3
4 from sklearn.metrics import classification_report
5
6 datafile_path = "data/fine_food_reviews_with_embeddings_1k.csv"
7
8 df = pd.read_csv(datafile_path)
9 df["embedding"] = df.embedding.apply(eval).apply(np.array)
10
11 # convert 5-star rating to binary sentiment
12 df = df[df.Score != 3]
13 df["sentiment"] = df.Score.replace({1: "negative", 2: "negative", 4: "positive",
```

每一条评论都用我们上面的方法，和一个预先设定好的好评和差评的文本去做对比，然后看它离哪个近一些。这里的好评和差评，我们写得稍微长了一点，分别是 “An Amazon review with a negative sentiment.” 和 “An Amazon review with a positive sentiment.”。

在计算完结果之后，我们利用 Scikit-learn 这个机器学习的库，将我们的预测值和实际用户打出的星数做个对比，然后输出对比结果。需要的代码，也就不到 20 行。

 复制代码

```
1 from sklearn.metrics import PrecisionRecallDisplay
2
3 def evaluate_embeddings_approach(
4     labels = ['negative', 'positive'],
5     model = EMBEDDING_MODEL,
6 ):
```



```

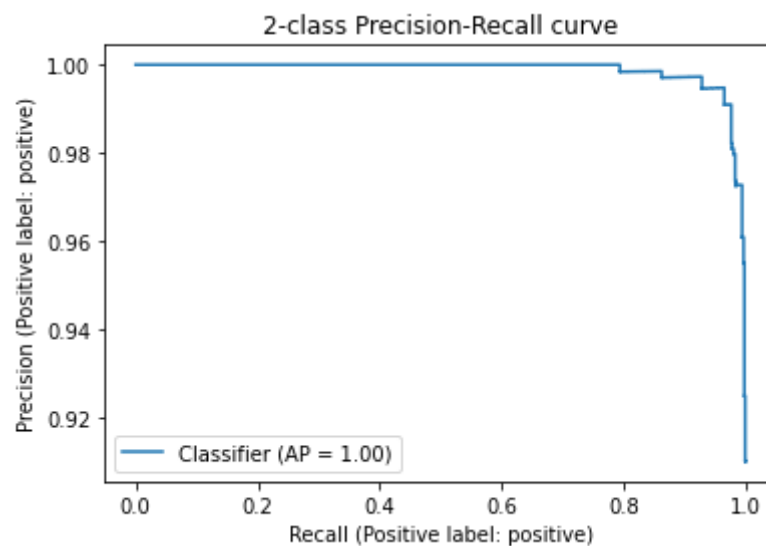
7     label_embeddings = [get_embedding(label, engine=model) for label in labels]
8
9     def label_score(review_embedding, label_embeddings):
10         return cosine_similarity(review_embedding, label_embeddings[1]) - cosine_
11
12     probas = df["embedding"].apply(lambda x: label_score(x, label_embeddings))
13     preds = probas.apply(lambda x: 'positive' if x>0 else 'negative')
14
15     report = classification_report(df.sentiment, preds)
16     print(report)
17
18     display = PrecisionRecallDisplay.from_predictions(df.sentiment, probas, pos_l
19     _ = display.ax_.set_title("2-class Precision-Recall curve")
20
21     evaluate_embeddings_approach(labels=['An Amazon review with a negative sentiment.

```

输出结果:

[复制代码](#)

		precision	recall	f1-score	support
2	negative	0.98	0.73	0.84	136
3	positive	0.96	1.00	0.98	789
4	accuracy			0.96	925
5	macro avg	0.97	0.86	0.91	925
6	weighted avg	0.96	0.96	0.96	925



在结果里面可以看到，我们这个简单方法判定的好评差评的精度，也就是 precision 在 negative 和 positive 里，分别是 0.98 和 0.96，也就是在 95% 以上。

而召回率，也就是图里的 recall，在差评里稍微欠缺一点，只有 73%，这说明还是有不少差评被误判为了好评。不过在好评里，召回率则是 100%，也就是 100% 的好评都被模型找到了。这样综合考虑下来的整体准确率，高达 96%。而要达到这么好的效果，我们不需要进行任何机器学习训练，只需要几行代码调用一下大模型的接口，计算一下几个向量的相似度就好了。

## 小结

这一讲，我们利用不同文本在大语言模型里 Embedding 之间的距离，来进行情感分析。这种使用大语言模型的技巧，一般被称做零样本分类（Zero-Shot Classification）。

所谓零样本分类，也就是我们不需要任何新的样本来训练机器学习的模型，就能进行分类。我们认为，之前经过预训练的大语言模型里面，已经蕴含了情感分析的知识。我们只需要简单利用大语言模型里面知道的“好评”和“差评”的概念信息，就能判断出它从未见过的评论到底是好评还是差评。

这个方法，在一些经典的数据集上，轻易就达到了 95% 以上的准确度。同时，也让一些原本需要机器学习研发经验才能完成的任务变得更加容易，从而大大降低了门槛。

如果你所在的公司今天想要做一个文本分类的应用，通过 OpenAI 的 API 用几分钟时间就能得到想要的结果。

## 课后练习

利用大模型来解决问题，不是一个理论问题，而是一个实战问题。请你尝试将这个方法运用在别的数据集上，看看是否也会有很好的效果？比如，你可以试着用 Kaggle 提供的亚马逊耳机类商品的评论数据，看看结果是怎样的？这个数据集比较大，你可以挑几条数据试一下。因为 OpenAI 会对你的调用限速，免费的 Token 数量也比较有限。对于限速情况下的大批量数据的处理，我们会在第 04 讲讲解。

数据集链接: <https://www.kaggle.com/datasets/shitalkat/amazonearphonesreviews>

欢迎你把你的结果分享到评论区, 也欢迎你把这节课分享给感兴趣的朋友, 我们下节课再见!

© 版权归极客邦科技所有, 未经许可不得传播售卖。页面已增加防盗追踪, 如有侵权极客邦将依法追究其法律责任。

## 精选留言 (67)



**Mr.x**

2023-04-07 来自北京

代码片段解释有ChatGPT标注

第二段代码:

```
from sklearn.metrics import PrecisionRecallDisplay
```

```
# 定义函数, 用于评估使用嵌入向量的情感分析模型
```

```
def evaluate_embeddings_approach(
```

```
    labels = ['negative', 'positive'], # 正负面情感标签列表, 默认为['negative', 'positive']
```

```
    model = EMBEDDING_MODEL, # 指定嵌入模型的变量, 默认为EMBEDDING_MODEL
```

```
):
```

```
    # 使用get_embedding函数获取每个情感标签的嵌入向量, 并存储在label_embeddings列表中
```

```
    label_embeddings = [get_embedding(label, engine=model) for label in labels]
```

```
    # 定义函数, 用于计算评价的情感得分
```

```
    def label_score(review_embedding, label_embeddings):
```

```
        # 计算评价嵌入向量与正面情感标签嵌入向量之间的余弦相似度, 减去评价嵌入向量与负面情感标签嵌入向量之间的余弦相似度
```

```
        return cosine_similarity(review_embedding, label_embeddings[1]) - cosine_similarity(review_embedding, label_embeddings[0])
```

```
    # 使用apply函数和lambda表达式计算每个评价的情感得分
```

```
    probas = df["embedding"].apply(lambda x: label_score(x, label_embeddings))
```

```
    # 将情感得分应用于阈值, 以确定每个评价的情感标签
```

```
preds = probas.apply(lambda x: 'positive' if x>0 else 'negative')

# 使用classification_report函数生成分类报告，并将其存储在变量report中
report = classification_report(df.sentiment, preds)
print(report)

# 使用PrecisionRecallDisplay函数生成精确率-召回率曲线，并在图表上设置标题
display = PrecisionRecallDisplay.from_predictions(df.sentiment, probas, pos_label='positive')
_ = display.ax_.set_title("2-class Precision-Recall curve")

# 调用函数，评估数据集的情感分析模型，并生成分类报告和精确率-召回率曲线的图表
evaluate_embeddings_approach(labels=['An Amazon review with a negative sentiment.', 'An Amazon review with a positive sentiment.'])
```



19



**渣渣辉**

2023-04-01 来自日本

之前浅学过深度学习技术，老师在这一章里把我一年学过的技术都给讲了还多讲了一些我不懂的技术！这就是大佬啊。



14



**humor**

2023-03-22 来自浙江

大模型是怎么知道好评和差评是代表什么的呢

作者回复：它是根据海量的预训练数据在高维空间里给他们两个找了个位置，我们只是我们拿到的评论离哪个位置近。

大模型的语义理解能力，就没法用一两句话说清了。如果真的想要知道，还是要深入理解深度学习，可以去看看李沐老师的论文解读系列。<https://www.bilibili.com/video/BV1AF411b7xQ/>

共 2 条评论 >

12



**Dev.lu**

2023-04-15 来自新加坡

```
conda install -c conda-forge matplotlib
conda install -c conda-forge plotly
conda install -c anaconda scikit-learn
```

跑情感分析的部分，可能需要额外安装这些包

作者回复: 👍

共 2 条评论 >

👍 12



**Roy Liang**

2023-03-22 来自浙江

课后作业，全部数据集是不是太多，出不了结果，前33条记录没问题，我是不是该给OpenAI充值？

	precision	recall	f1-score	support
negative	0.81	0.93	0.87	14
positive	0.92	0.79	0.85	14
accuracy			0.86	28
macro avg	0.86	0.86	0.86	28
weighted avg	0.86	0.86	0.86	28

代码

```
import pandas as pd
import numpy as np
```

```
from sklearn.metrics import classification_report, PrecisionRecallDisplay
from openai.embeddings_utils import cosine_similarity, get_embedding
```

```
EMBEDDING_MODEL = "text-embedding-ada-002"
```

```
#datafile_path = "data/AllProductReviews.csv"
datafile_path = "data/test.csv"
df = pd.read_csv(datafile_path)
df["embedding"] = df.ReviewBody.apply(lambda x: get_embedding(x, engine=EMBEDDING_MODEL))
```

```

# convert 5-star rating to binary sentiment
df = df[df.ReviewStar != 3]
df["sentiment"] = df.ReviewStar.replace({1: "negative", 2: "negative", 4: "positive", 5: "positive"})

def evaluate_embeddings_approach(
    labels = ['negative', 'positive'],
    model = EMBEDDING_MODEL,
):
    label_embeddings = [get_embedding(label, engine=model) for label in labels]

    def label_score(review_embedding, label_embeddings):
        return cosine_similarity(review_embedding, label_embeddings[1]) - cosine_similarity(review_embedding, label_embeddings[0])

    probas = df["embedding"].apply(lambda x: label_score(x, label_embeddings))
    preds = probas.apply(lambda x: 'positive' if x>0 else 'negative')

    report = classification_report(df.sentiment, preds)
    print(report)

    display = PrecisionRecallDisplay.from_predictions(df.sentiment, probas, pos_label='positive')
    _ = display.ax_.set_title("2-class Precision-Recall curve")

evaluate_embeddings_approach()

```

作者回复: 这个是API限速导致的, 我们在第4讲里会讲解应该怎么处理。

这个思考题看来是我考虑不周了, 我建议拿个100条试一下吧, 全量数据可以看完第四讲之后再来, 而且需要注意Token费用问题。

共 4 条评论 >

👍 8



jo

2023-04-07 来自上海

“我们把这段文本的 Embedding 和“好评”以及“差评”通过余弦距离（Cosine Similarity）计算出它的相似度”，不太懂

作者回复: 不理解什么是向量的余弦距离, 可以问问ChatGPT:

什么是向量的余弦距离?

向量的余弦距离是用于度量两个向量之间的相似性的一种方法。它衡量的是两个向量之间的夹角的余弦值。

具体来说, 假设有两个n维向量a和b, 其余弦距离为 $\cos(\theta)$ , 其中 $\theta$ 是a和b之间的夹角, 计算公式如下:

$$\cos(\theta) = (a \cdot b) / (||a|| \ ||b||)$$

其中,  $a \cdot b$ 表示向量a和向量b的点积,  $||a||$ 和 $||b||$ 分别表示向量a和向量b的范数。

余弦距离的取值范围在 $[-1, 1]$ 之间, 当两个向量之间的夹角越小时, 余弦距离越接近1, 表示两个向量越相似; 当两个向量之间的夹角越大时, 余弦距离越接近-1, 表示两个向量越不相似。当夹角为90度时, 余弦距离为0, 表示两个向量完全不相关。

共 2 条评论 >

👍 6



**Mr.x**

2023-04-07 来自北京

代码片段解释有ChatGPT标注

代码片段一:

```
import pandas as pd
import numpy as np
```

```
from sklearn.metrics import classification_report
```

```
# 设置数据集文件路径
```

```
datafile_path = "data/fine_food_reviews_with_embeddings_1k.csv"
```

```
# 使用pandas的read_csv函数读取csv文件, 并将其存储在名为df的数据框中
```

```
df = pd.read_csv(datafile_path)
```

```
# 使用apply函数和lambda表达式, 将embedding列中的每个字符串转换为Python对象, 并将其转换为NumPy数组
```



```
df["embedding"] = df.embedding.apply(eval).apply(np.array)
```

```
# 将评分为3的评价从数据框中删除
```

```
df = df[df.Score != 3]
```

```
# 将评分为1、2的评价标记为negative，将评分为4、5的评价标记为positive，生成一个新的sentiment列
```

```
df["sentiment"] = df.Score.replace({1: "negative", 2: "negative", 4: "positive", 5: "positive"})
```



👍 4



**陈敏俊**

2023-03-22 来自上海

文本向量positive和negative标签是openai训练标注好的，那还有其他什么标签吗？只有这两种吗？

作者回复：这不是预先训练好的标签。而是模型根据文本，计算出来的向量，我们只是利用了向量的相似度而已。

你可以用任意的单词，或者句子来做这样的功能，比如 beautiful 和 ugly，polite 和 rude 都可以试一试

共 3 条评论 >

👍 4



**Bing**

2023-03-22 来自加拿大

受教了，谢谢！



👍 4



**xbc**

2023-03-22 来自海南

后续是否会讲讲如何使用openai api实现chatpdf这样的功能。以及使用公开的弱一些模型，来尽量实现chatpdf的功能。

作者回复：专栏的第二部分，实战应用部分里，会大量涵盖这部分内容。也会通过一些开源模型来降低使用的成本。

**陈鹏**

2023-04-28 来自立陶宛

用了亚马逊耳机数据集，取前30条数据，结果如下，如果用excel打开csv文件，好像会导致报UnicodeDecodeError 错误

	precision	recall	f1-score	support
negative	1.00	0.67	0.80	3
positive	0.80	1.00	0.89	4
accuracy			0.86	7
macro avg	0.90	0.83	0.84	7
weighted avg	0.89	0.86	0.85	7



👍 2

**钱雨**

2023-04-04 来自河北

经历了无数次下面的失败，不停重新安装pillow和matplotlib之后，终于成功了。  
 解决方案是狠心把miniconda、python全卸载，重新安装。  
 注意install openai以后，默认的pillow版本是9.4，太高了，把pillow卸载，安装8.4。  
 这么操作以后，新的问题是openai又没了，就把openai重新装回来。  
 很开心的是，pillow居然还是8.4.结果正常了。

File ~\miniconda3\envs\py310\lib\site-packages\PIL\Image.py:103

```
94 MAX_IMAGE_PIXELS = int(1024 * 1024 * 1024 // 4 // 3)
```

```
97 try:
```

```
98     # If the _imaging C module is not present, Pillow will not load.
```

```
99     # Note that other modules should not refer to _imaging directly;
```

```
100     # import Image and use the Image.core variable instead.
```

```
101     # Also note that Image.core is not a publicly documented interface,
```

```
102     # and should be considered private and subject to change.
```

```
--> 103     from . import _imaging as core
```

```
105     if __version__ != getattr(core, "PILLOW_VERSION", None):
```

```
106         msg = (
```

```
107             "The _imaging extension was built for another version of Pillow or PIL:\n"
```

```
108             f"Core version: {getattr(core, 'PILLOW_VERSION', None)}\n"
```

```
109             f"Pillow version: {__version__}"
```

ImportError: DLL load failed while importing \_imaging: The specified module could not be found.

作者回复: python的环境的确是个问题, 我看一下是否晚点能搞个Docker的Image来解决问题

共 2 条评论 >



2



**渣渣辉**

2023-04-02 来自日本

为什么同样是用chatgpt的出来的结果跟老师的事例稍有不同, 是因为chatgpt这个模型是在实时更新的吗

作者回复: 有两个原因

1. chatgpt的模型在实时更新, 我选用的一般都不是快照的模型, 而是模型最新版本。
2. 本身 GPT 输出结果是概率采样, 有一定的随机性。



2



**piboye**

2023-03-22 来自广东

老师, text complete 和 chat complete 接口有什么区别啊, 我看能力没看出大的差别

作者回复: Chat Complete是针对对话微调过的模型。我自己的体验是很多时候给它明确的指令, 它也会有点话唠, 废话有点多。

共 3 条评论 >



2



**peter**

2023-03-22 来自山西

请教老师几个问题:

Q1: 01课中的Jupyter Labs 开发环境是python的开发环境吗? (对python不了解, 0基础, 小白)

Q2: 01课中的Jupyter Labs 开发环境搭建, 复杂吗? 如果复杂的话, 能否麻烦老师写一个手册一类的文档? 最好详细一点。(一直觉得搭环境很麻烦; 尤其对不熟悉的python, 容易出错, 容易有挫折感。)。环境搭不好, 后面的例子无法实践。

Q3: Cookbook例子, 为什么用Scikit估计结果, 应该都用openai来完成啊。例子中用Scikit,

那相当于“openai + Scikit”结合来判断出最后的星标了，openai这么强大，还需要用Scikit吗？

Q4: chatGPT搜索或生成ppt的能力如何？

比如我要做一个软件的产品介绍性质的ppt，用来给客户介绍该软件。以matlab为例，我托国外同事这样搜“matlab 产品介绍 ppt”，并没有搜到合适的ppt，后来只好在百度上搜索了。请问，是我们的使用方法不对？还是说chatGPT没有这方面的能力？

作者回复: Q1: 是的，一个交互式可以看到输出结果，图片等各种功能的开发环境

Q2: 可以直接使用在线的Colab，就不需要自己搭建了，可以看一下第一讲。搭建的复杂程度每个人感觉不一样。

Q3: 并不是所有的问题都适合用OpenAI的API来完成。我们需要考虑成本，OpenAI的Embedding接口很便宜，但是Completion接口就会贵一些（即使是gpt-3.5-turbo），也要考虑效果，有标注数据的情况下，利用标注会比只用大模型效果更好

Q4: 制作PPT，我们可以期待一下微软最新的发布会里介绍的 Office Copilot，你只要给它合适的资料，可以自动生成PPT。你可以试一下 beta.tome.app 是一个创业工作做的通过AI创建PPT的产品。

共 2 条评论 >

👍 2



**Geek\_035566**

2023-05-08 来自广东

知乎上张俊林对GPT对NLP的影响做了很好解读，我这个小白听了也能有很深的理解。

他把NLP解决的问题分为中间任务和最终任务，先阐述中间任务其实是人类没有找到更好办法处理自然语言时想出来的辅助工具，GPT的出现一瞬间让这些脚手架都没了用处。

再进一步，他又把最终任务分为自然语言理解和自然语言生成任务，进一步分析GPT对这两类任务的影响。

很简洁优美的讲清楚了GPT对NLP的冲击，知乎地址我放下面，感兴趣的可以自己看。

<https://zhuanlan.zhihu.com/p/597586623>

作者回复: 👍

共 2 条评论 >

👍 1



**Geek\_b7449f**

2023-05-06 来自美国

```
import pandas as pd
import numpy as np
```

```

from sklearn.metrics import classification_report , PrecisionRecallDisplay
from openai.embeddings_utils import cosine_similarity , get_embedding

# 选择用于情感分析的模型
EMBEDDING_MODEL = "text-embedding-ada-002"

# 设置数据集文件路径
datafile_path = "data/AllProductReviews.csv"

# 读取 csv 文件 只取前60行
df = pd.read_csv(datafile_path,nrows=60)

# 新增一列 embedding 用于表示获取每行的ReviewBody向量值
df["embedding"] = df.ReviewBody.apply(lambda x:get_embedding(x,engine=EMBEDDING_MODEL))

# 除去星级为3的评论来进行划分好评还是差评两个标签
df = df[df.ReviewStar != 3]
df["sentiment"] = df.ReviewStar.replace({1: "negative", 2: "negative", 4: "positive", 5: "positive"})

# 函数 evaluate_embeddings_approach 默认会有两个参数 label 和 model 都有对应的默认值
def evaluate_embeddings_approach(
    labels = ['negative', 'positive'],
    model = EMBEDDING_MODEL,
):
    # 计算每个 label 的向量值
    label_embeddings = [get_embedding(label, engine=model) for label in labels]
    # 辅助函数 用于计算好评和review_embedding与差评和review_embedding向量余弦值差值
    def label_score(review_embedding, label_embeddings):
        return cosine_similarity(review_embedding, label_embeddings[1]) - cosine_similarity(review_embedding, label_embeddings[0])
    # 这里更新下 embedding 列 更新成对应的 分数
    probas = df["embedding"].apply(lambda x: label_score(x, label_embeddings))
    # 预测值
    preds = probas.apply(lambda x: 'positive' if x>0 else 'negative')
    # 生成一个分类报告
    report = classification_report(df.sentiment, preds)
    print(report)
    # 这个就是绘制二分类的准确率-召回率曲线

```

```
display = PrecisionRecallDisplay.from_predictions(df.sentiment, probas, pos_label='positive')
_ = display.ax_.set_title("2-class Precision-Recall curve")
# 无参数默认值调用函数
evaluate_embeddings_approach()
```

作者回复: 👍



👍 1



**漫游者**

2023-05-05 来自北京

```
negative_example = get_embedding("随意降价，不予价保，服务态度差")
```

差评例子的评分：-0.081472

```
negative_example = get_embedding("随意降价，不予价保，服务态度差，但是后续换了个客服，服务态度好多了")
```

差评例子的评分：-0.048267

```
negative_example = get_embedding("随意降价，不予价保，服务态度差，但是后续换了个客服，服务态度好多了。经过沟通解决了所有问题，好评！")
```

差评例子的评分：-0.009858

通过例子看到，第三句话很明显我已经是好评了，但是模型还是给了差评的结论。是不是排在前面的句子权重更大啊？

作者回复: 不能简单根据内容排在前面后面来判定。

这样的例子本身的确人来判断也很难明确定义算好还算坏，从分数来看也比另外两个更接近0.



👍 1



**lhs**

2023-04-17 来自福建

这个调openapi提供的api进行训练数据，应该要考虑数据安全问题吧?比如用公司真实的业务数据进行训练，可能会发生数据泄露问题

作者回复: 合规就好了，OpenAI协议写明了不会使用API提交的数据用于模型训练。



**aoe**

2023-03-24 来自浙江

看到这节课，我才知道老师为什么不使用 Playground 了，API 能做更多事情

