文本表征学习 Lab 3 Doc2vec

PB22111599 杨映川

Pt.1 实验概述

使用IMDB情感分析数据集（包括训练集和测试集），采取下面四种算法训练doc2vec向量，并基于逻辑回归算法在测试集上对四个模型的性能进行评估。

- **HS + PV-DM**
- **HS + PV-DBOW**
- **NS + PV-DM**
- **NS + PV-DBOW**

> - HS: Hierarchical Softmax;
> - NS: Negative Sample;
> - PV-DM: Distributed Memory Model of Paragraph Vectors;
> - PV-DBOW: Distributed Bag of Words version of Paragraph Vector

Pt.2 数据处理与实现过程

2.1 模型训练

IMDB数据集可以通过安装python库`datasets`后下载使用。通过`gensim.models.doc2vec.Doc2vec`模型可以将标记好的数据集训练成向量模型。查阅gensim官方文档可知Doc2vec类继承了Word2vec类，故可以使用后者的参数。

```
158    class Doc2Vec(Word2Vec):
159        def __init__(
160                self, documents=
```

> 参数
>
> - vector_size=200 # 向量维度
> - window=5 # 窗口大小
> - min_count=5 # 词频过滤
> - dm=1 or 0 # 使用PV-DM算法或PV-DBOW算法
> - negative=20 or 0 # 负采样20个或不进行负采样
> - hs=1 or 0 # 是否使用层次化softmax
> - epochs=20 # 模型迭代次数

2.2 模型评估

使用`sklearn.linear_model.LogisticRegression`工具作为逻辑回归模型，在测试集上对比预测的结果与实际结果，并计算出准确率。定义评估函数`evaluate_model`函数如下：

```python
def evaluate_model(train_vectors, test_vectors, train_labels, test_labels):
    """Train and evaluate a logistic regression model."""
    classifier = LogisticRegression(max_iter=1000)
    classifier.fit(train_vectors, train_labels)
    predictions = classifier.predict(test_vectors)
    accuracy = accuracy_score(test_labels, predictions)
    return accuracy
```

对于逻辑回归模型的参数选取，出于训练数据集较小的考虑，使用了penalty=None。（实际测试后发现penalty参数的值对结果影响很小，故统一采用无惩罚机制）

Pt.3 训练结果

分类准确率（%）	HS	NS
PV-DM	81.96(0.032)	83.19(0.021)
PV-DBOW	85.78(0.011)	85.54(0.022)

- 使用相同参数分别进行了4次训练和测试，取四次结果的平均数。括号内为方差。

Pt.4 分析

1. 在测试集上PV-DBOW的性能要比PV-DM的效果更好；
2. 优化策略上，在使用PV-DM算法时，NS的表现要优于HS的表现；而在使用PV-DBOW算法时，NS和HS没有表现出明显的差异，HS算法在很微小的程度上优于NS算法（可以忽略）；
3. 使用NS优化策略时，模型的训练会花费比HS更多的时间（训练时长相差近一倍）。推测是因为NS采样了较多数据（20个）。推测2)中的第一条结论也得益于此；
4. 四种组合的方差都很小，说明模型相对比较稳定；

Pt.5 其他

1. 迭代次数epochs=20，完整训练一次所有模型并测试约花费1小时。训练模型约10分钟，将模型转化成可比的向量约50分钟，回归评估时间少于1秒；
2. 此次实验在conda提供的虚拟环境下完成，附上所有使用的库

```python
from datasets import load_dataset
from gensim.models import Doc2Vec
from gensim.models.doc2vec import TaggedDocument
from gensim.utils import simple_preprocess
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
import time
```