NLP情感分析实验报告

韩方朔2020201405

实验背景及目的

情感分析作为自然语言处理的重要组成部分，广泛应用于网络舆情、服务评价等现实场景中，因此，本实验通过构建体系化的数据处理、文本挖掘、词嵌入等NLP常用方法理论，实现中文段落的情感分类，并采用GaussianNB、LogisticRegression、LinearSVC等聚类算法进行交叉验证。

实验数据

本实验基于分类的文本数据，即每段文本对应一个情感标签（正向和负向），数据可以来源于网络舆情监控、服务评价等领域。随报告提交的代码中，选取的是来自某平台的酒店服务评价，包括7000条正向评价和3000条负向评价，分别保存在dataset\pos.txt和dataset\neg.txt中。

数据处理

首先对中文段落文本进行分词处理，结合jieba的分词功能和停用词库。

```
def tokenization(doc, sw):
    """
    分词
    :param doc:          文本内容
    :param sw:           停用词列表
    :return:             分词后的列表
    """
    result = []
    text = re.findall('[\u4e00-\u9fa5]', doc)  # 保留汉字
    words = pseg.cut(''.join(text))
    for word, flag in words:
        if flag not in stop_flag and word not in sw:
            result.append(word)
    return ' '.join(result)
```

分词结果保存在dataset/words_w2v.txt中，用于后续的词向量生成。

词向量生成

在得到分词后的文本后，需要进一步将词语量化，转为计算机可识别语言。本实验采用word2vec模块提供的词向量映射。

```
class W2v(object):
    def __init__(self,fin=None):
        self.model = None
        self.file = fin

    def load(self,fmodel):
        self.model = word2vec.Word2Vec.load(fmodel)
```

```
    def process(self,size=100,window=5,min_count=5):
        import multiprocessing #并行化
        cpu_count = multiprocessing.cpu_count()
        workers = cpu_count
        sentences = word2vec.LineSentence(self.file)
        self.model =
word2vec.Word2Vec(sentences,vector_size=size,window=window,min_count=min_count,worke
rs=workers)

    def save(self,fout):
        try:
            self.model.save(fout)
```

得到的w2v模型保存在dataset/w2v.model中。

特征分类

本实验采用三种特征分类方法，分别为Word2vec，TF-IDF和LSI特征。

Word2vec特征

word2vec 是词嵌入（word embedding）的一种，它会将一个词映射到一个固定维度的向量中（不随语料的变化而变化），并且能够在一定程度上反映出词与词之间的关系。

借助上步实现的w2v模型，可以得到如下代码实现：

```
    from gensim.models import word2vec
    w2vmodel = word2vec.Word2Vec.load('.//dataset//w2v.model')
    ct = []
    for row in df[parm_data].iteritems():
        t = []
        for word in row[1].split(' '):
            if word in w2vmodel.wv.key_to_index:
                t.append(w2vmodel.wv[word])
        ct.append(np.mean(np.array(t), axis=0).tolist())
    df['w2v'] = ct
```

TF-TDF特征

“词频 - 逆向文件频率”（TF-IDF）是一种在文本挖掘中广泛使用的特征向量化方法，它可以体现一个文档中词语在语料库中的重要程度。

词语由t表示，文档由d表示，语料库由D表示。词频TF(t,d)是词语t在文档d中出现的次数。文件频率DF(t,D)是包含词语的文档的个数。TF-IDF 度量值表示如下：

$$TFIDF(t,d,D) = TF(t,d) \cdot IDF(t,D)$$

根据sklearn提供的TF-IDF模型，得到生成函数：

```
    tfidf = TfidfVectorizer(
        analyzer='word',
        ngram_range=(1, 4),
        max_features=150000
    )
```

LSI特征

潜在语义索引(LSI)，又称为潜在语义分析(LSA)，主要为了解决两类问题，一类是一词多义问题，另一类是一义多词问题。潜在语义索引利用SVD降维的方法将词项和文本映射到一个新的空间。

实现时，在TF-IDF模型的基础上，进行SVD降维处理。

```
    tfidf = TfidfVectorizer(
        analyzer='word',
        ngram_range=(1, 4),
        # max_features=150000
    )
    # tfidf.fit(data_)
    x_train = tfidf.fit_transform(data_)

    lsa = TruncatedSVD(n_components=300, n_iter=10, random_state=42)
    # lsa.fit(x_train)
    x_train = lsa.fit_transform(x_train)
```

交叉验证

本实验交叉验证采用10折交叉的方式，分别采用贝叶斯，LR，LinearSVC单重聚类算法进行验证。

```
    model = GaussianNB()
    accs = sk_model_selection.cross_val_score(model, x_train, y=label,
scoring='precision', cv=10, n_jobs=1)
    print('GaussianNB 交叉验证结果:', accs, '均值：', accs.mean())

    model = LogisticRegression()
    accs = sk_model_selection.cross_val_score(model, x_train, y=label,
scoring='precision', cv=10, n_jobs=1)
    print('LogisticRegression 交叉验证结果:', accs,'均值：', accs.mean())

    model = LinearSVC()
    accs = sk_model_selection.cross_val_score(model, x_train, y=label,
scoring='precision', cv=10, n_jobs=1)
    print('LinearSVC 交叉验证结果:', accs, '均值：', accs.mean())
```

实验结果

验证\算法	Word2vec	TF-IDF	LSI
贝叶斯验证	0.821	0.703	0.832
LR验证	0.809	0.806	0.779
LinearSVC验证	0.824	0.863	0.861

实验数据结果表明，word2vec词向量算法和LSI潜在语义索引在该数据集下效果较好，其中Word2vec算法在多组验证中效果最为均衡。

实验总结

本实验设计了一个通用的中文情感分析实现流程，其中特征分类、交叉验证均给出多个可替换的策略，同时对数据集的要求比较低（仅要求给出分类标签）。该模型可应用在包括舆情监测、评论分析等多个领域。

附测试记录

word2vec：

```
 GaussianNB 交叉验证结果: [0.8018648  0.81861575 0.84047619 0.79949239 0.82382134 0.81508516
  0.84810127 0.84810127 0.81       0.80645161] 均值:  0.8212009772567791
 LogisticRegression 交叉验证结果: [0.80936455 0.81063123 0.81438127 0.80677966 0.8159204  0.79452055
  0.81525424 0.81727575 0.80065898 0.80681818] 均值:  0.8091604800803799
 LinearSVC 交叉验证结果: [0.82504288 0.83076923 0.82847341 0.81989708 0.82799325 0.81382979
  0.84042553 0.82372881 0.8163606  0.81713344] 均值:  0.8243654041353448
```

TF_IDF：

```
MultinomialNB 交叉验证结果: [0.69411765 0.70478723 0.70026525 0.70238095 0.70627503 0.70493992
 0.7014531  0.71028037 0.70424403 0.70277411] 均值:  0.70315176570864
LogisticRegression 交叉验证结果: [0.80690738 0.7953125  0.79658385 0.81072555 0.80841121 0.81064163
 0.80974843 0.81875994 0.80438185 0.80337942] 均值:  0.8064851750812367
LinearSVC 交叉验证结果: [0.86538462 0.84812287 0.86842105 0.8554007  0.86643234 0.86481802
 0.86643234 0.86910995 0.86701209 0.86026936] 均值:  0.8631403328629481
```

LSI：

```
GaussianNB 交叉验证结果: [0.82587065 0.85026738 0.8543956  0.82384824 0.82507289 0.81967213
 0.83050847 0.85337243 0.82978723 0.80813953] 均值:  0.8320934564288359
LogisticRegression 交叉验证结果: [0.78463855 0.78333333 0.77361319 0.77627628 0.76843658 0.76740741
 0.78614458 0.7869102  0.78419453 0.77695716] 均值:  0.778791181182465
LinearSVC 交叉验证结果: [0.85615251 0.85061511 0.8492201  0.84859155 0.85964912 0.85638298
 0.88703704 0.87697715 0.87346221 0.85092127] 均值:  0.8609009059426034
```