71086032-曾诗仪-第五周作业

在进行自然语言处理时，常常会发现在某个特定的语言环境或者平台内可能会存在很多不在我们python情绪词典中，但也可以用来表示情绪，并且使用频率相当大的词语，例如“温油”、“亮眼”等。Word2vec是一类用来产生词向量的神经网络模型，通过学习文本语料库中单词之间的语义和上下文关系来表示单词的向量空间。在训练完成后，它可以把每个词映射到一个可以表示词与词之间关系的向量。因此它可以被用来进行文本聚类和相似度计算等任务，也常常被应用在包括文本分类、情感分析、信息检索、机器翻译等场景下。因此，本次作业用实现一个简单的类，并体会word2vec的各种相关应用。

1. 定义一个类TextAnalyzer，其属性包括待分析的文本文件路径，等加载的预训练模型文件路径，训练word2vec的一些简单参数（如向量长度，窗口大小）等，初始化的时候需要对这些属性进行定义。

```
model = Word2Vec(sentences= sentences, vector_size=500, alpha=0.025,
window=5,
                                 min_count=1, max_vocab_size=None, sample=1e-3,
seed=1, workers=3, min_alpha=0.0001, sg=0,
                                 hs=0, negative=5, ns_exponent=0.75,
cbow_mean=1, hashfxn=hash, epochs=5, null_word=0,
                                 trim_rule=None, sorted_vocab=1,
batch_words=MAX_WORDS_IN_BATCH, compute_loss=False,
                                 callbacks=(), comment=None,
max_final_vocab=None, shrink_windows=True)
```

2. 在上述类加入一些方法，如将待分析的weibo.txt加载到内存，进行基本的文本预处理，如对所有微博进行预处理，包括分词、去除停用词、标点等，最终建立一个以微博为单位进行分词的二维列表。

```
import jieba
from sklearn.manifold import TSNE
from gensim.models import Word2Vec
import matplotlib.pyplot as plt
from matplotlib.font_manager import FontProperties
sinmei = FontProperties(fname='C:\\Users\\shiye\\Desktop\\simhei.ttf')
import numpy as np
MAX_WORDS_IN_BATCH=1000


# 定义语料库（省略文本预处理）
stopwords = set(['[',']','@'])
sentences = []
with open('C:\\Users\\shiye\\Desktop\\python\\weibo.txt','r', encoding='utf-
8') as f:
    for line in f:
        vocab = [w for w in jieba.cut(line.strip().split('\t')[1]) if w not
in stopwords]
        sentences.append(vocab)
print(f'load {len(sentences)} tweets...')
```

3. 在上述类加入一个方法来利用weibo.txt中的文本构建Word2Vec模型model。使用gensim完成Word2Vec模型的建立。

```
model = Word2Vec(sentences= sentences, vector_size=500, alpha=0.025,
window=5,
                                 min_count=1, max_vocab_size=None, sample=1e-3,
seed=1, workers=3, min_alpha=0.0001, sg=0,
                                 hs=0, negative=5, ns_exponent=0.75,
cbow_mean=1, hashfxn=hash, epochs=5, null_word=0,
                                 trim_rule=None, sorted_vocab=1,
batch_words=MAX_WORDS_IN_BATCH, compute_loss=False,
                                 callbacks=(), comment=None,
max_final_vocab=None, shrink_windows=True)
```

4. 在上述类加入一个方法来利用训练得以的word2vec模型来推断相似词汇。如输入一个任意词，使用model.similarity方法可以来返回与目标词汇相近的一定数目的词。

```
# 获取最相关的10个词和最不相关的10个词
most_similar = model.wv.most_similar("美丽", topn=10)
least_similar = model.wv.most_similar(negative=["美丽"], topn=10)

print(most_similar)
print(least_similar)

# 模型的保存
model.save("word2vec.model")
```

5. 加载预训练模型。训练所得的word2vec模型可以提供给其他人使用。请在类中增加一个方法，来加载预训练模型（如本次作业会提供的weibo_59g_embedding_200. model），并通过相似词的差异来比较分析一下与3中训练的word2vec模型的差异。

```
# 预训练模型的加载
model = Word2Vec.load("word2vec.model")
print(model.wv.most_similar("美丽", topn=5))

# 将最相关和最不相关的词汇向量合并为一个数组
vec = np.array([model.wv[word] for word, similarity in most_similar +
least_similar])
print(vec.shape)
words = [word for word, similarity in most_similar + least_similar]
```

6. （附加）这些相似词可否用来扩展我们之前使用过的情感词典？请在类中再增加一个方法，对情感词典进行扩充，如对于情感词典中已有的人工标定的词，找到与其相似的词（如top 5），标记为同样的情感标签，并加入到词典，观察其对情绪分类是否有提升作用（比如增加了覆盖率等）。

7. （附加）词向量能够将每个词定位为高维空间中的一个点，且不同词间的“差异”可以通过点间的距离来反应。在类加再增加一个方法，使用sklearn.manifold中的TSNE算法，对与某个输入词的最相关以及最不相关的词语（用参数来控制数目）进行降维和可视化。

```
# 使用t-SNE算法对词向量进行降维
tsne = TSNE(n_components=2, perplexity=10)
vectors_tsne = tsne.fit_transform(vec)

# 可视化降维后的词向量
fig, ax = plt.subplots()
ax.set_title('美丽', fontproperties = sinmei)
ax.scatter(vectors_tsne[:10, 0], vectors_tsne[:10, 1], color='blue')
ax.scatter(vectors_tsne[10:, 0], vectors_tsne[10:, 1], color='red')
for i, word in enumerate(words):
    ax.annotate(word, (vectors_tsne[i, 0], vectors_tsne[i,
1]),fontproperties = sinmei)
plt.show()
```

8. 参考资料

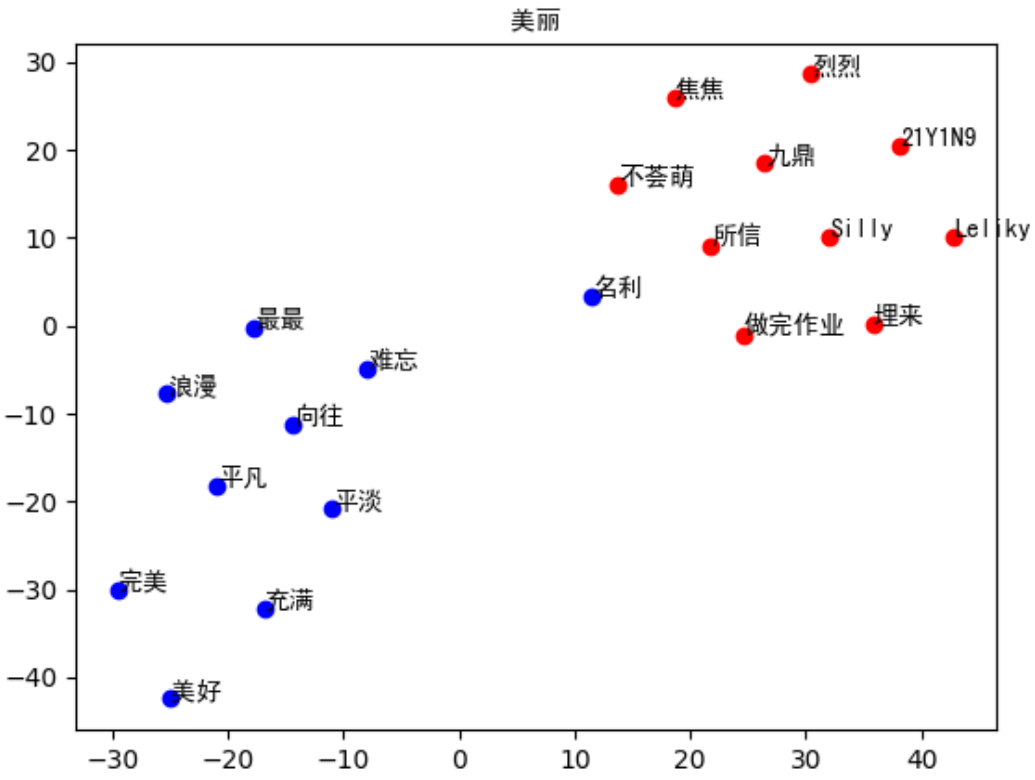
   https://radimrehurek.com/gensim/models/word2vec.html
   https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html

完整代码：

```
import jieba
from sklearn.manifold import TSNE
from gensim.models import Word2Vec
import matplotlib.pyplot as plt
from matplotlib.font_manager import FontProperties
sinmei = FontProperties(fname='C:\\Users\\shiye\\Desktop\\simhei.ttf')
import numpy as np
MAX_WORDS_IN_BATCH=1000


# 定义语料库（省略文本预处理）
stopwords = set(['[',']','@'])
sentences = []
with open('C:\\Users\\shiye\\Desktop\\python\\weibo.txt','r', encoding='utf-8')
as f:
    for line in f:
        vocab = [w for w in jieba.cut(line.strip().split('\t')[1]) if w not in
stopwords]
        sentences.append(vocab)
print(f'load {len(sentences)} tweets...')

# 建立Word2Vec模型
#model = Word2Vec(sentences, vector_size=500, window=5, min_count=1)
model = Word2Vec(sentences= sentences, vector_size=500, alpha=0.025, window=5,
```

```
                                min_count=1, max_vocab_size=None, sample=1e-3,
seed=1, workers=3, min_alpha=0.0001, sg=0,
                                hs=0, negative=5, ns_exponent=0.75, cbow_mean=1,
hashfxn=hash, epochs=5, null_word=0,
                                trim_rule=None, sorted_vocab=1,
batch_words=MAX_WORDS_IN_BATCH, compute_loss=False,
                                callbacks=(), comment=None, max_final_vocab=None,
shrink_windows=True)
# 获取最相关的10个词和最不相关的10个词
most_similar = model.wv.most_similar("美丽", topn=10)
least_similar = model.wv.most_similar(negative=["美丽"], topn=10)

print(most_similar)
print(least_similar)

# 模型的保存
model.save("word2vec.model")

# 预训练模型的加载

model = Word2Vec.load("word2vec.model")
print(model.wv.most_similar("美丽", topn=5))

# 将最相关和最不相关的词汇向量合并为一个数组
vec = np.array([model.wv[word] for word, similarity in most_similar +
least_similar])
print(vec.shape)
words = [word for word, similarity in most_similar + least_similar]

# 使用t-SNE算法对词向量进行降维
tsne = TSNE(n_components=2, perplexity=10)
vectors_tsne = tsne.fit_transform(vec)

# 可视化降维后的词向量
fig, ax = plt.subplots()
ax.set_title('美丽', fontproperties = sinmei)
ax.scatter(vectors_tsne[:10, 0], vectors_tsne[:10, 1], color='blue')
ax.scatter(vectors_tsne[10:, 0], vectors_tsne[10:, 1], color='red')
for i, word in enumerate(words):
    ax.annotate(word, (vectors_tsne[i, 0], vectors_tsne[i, 1]),fontproperties =
sinmei)
plt.show()
```

输出结果


美丽
烈烈
焦焦
21Y1N9
九鼎
不荟萌
所信
Silly
Leliky
名利
最最
做完作业
埋来
难忘
浪漫
向往
平凡
平淡
完美
充满
美好
30
20
10
0
−10
−20
−30
−40
−30
−20
−10
0
10
20
30
40