71086032 曾诗仪 第四周作业

LDA（Latent Dirichlet allocation）是一种常用的主题模型，它可以将文档集中每篇文档的主题按照概率分布的形式给出。它是一种无监督学习算法，在训练时仅需要输入文档集并给定主题数量。这一模型目前在文本挖掘，包括文本主题识别、文本分类以及文本相似度计算方面均有应用。请利用weibo.txt数据，进一步对微博文本进行话题识别与分析。

1. 文档的生成。一般来讲，LDA在微博等短文本上的效果并不理想 ，且多数情况下，我们希望给话题赋予时间含义，以讨论其"波动性"。因此，往往先需要按时间进行文档的生成 ，比如，将微博将天进行合并 ，即相同一天发布的所有微博，视为一个文档。请实现一个模块，其中包含一个或多个函数，其能够读取weibo.txt并将之处理以天（或其他时间单位）为单位的文档集合。
    - 需要将weibo.txt文件按天进行合并，将同一天发布的所有微博视为一个文档。可以先读取weibo.txt文件，逐行读取每个微博的发布时间和内容，将它们保存到一个列表中。然后根据每个微博的发布时间，将它们分成不同的天，并将同一天的微博内容合并成一个文档。最终返回一个列表，其中每个元素为一天的微博文档。

```
import re
import jieba
import datetime
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.decomposition import LatentDirichletAllocation
import pickle

def sort_doc(file_path):
    # 定义时间段
    morning_start = datetime.time(4, 0, 0)
    noon_start = datetime.time(12, 0, 0)
    evening_start = datetime.time(20, 0, 0)

    # 定义数据分类字典
    data_by_time = \
        {
        "morning": [],
        "noon": [],
        "evening": []
        }

    # 读取数据文件
    with open(file_path, "r", encoding = "utf-8") as f:
        file_path:"C:/Users/shiye/Desktop/python/cn_stopwords.txt"
    for line in f:
        # 解析每一行数据
        fields = line.strip().split("\t")
        coords = fields[0]
        text = fields[1]
        time_str = fields[2]

        # 将时间字符串转换为datetime对象
        time = datetime.datetime.strptime(time_str, "%a %b %d %H:%M:%S %z
%Y").time()
```

```
        # 根据时间段将数据添加到相应的分类列表中
        if time >= morning_start and time < noon_start:
            data_by_time["morning"].append((coords, text))
        elif time >= noon_start and time < evening_start:
            data_by_time["noon"].append((coords, text))
        else:
            data_by_time["evening"].append((coords, text))
            return data_by_time
```

2. 文本的预处理。文本数据的预处理往往能够决定分析任务的成败。请实现一个模块，其中包含一个或多个函数，能够对所有文档进行预处理，包括分词、去除停用词、标点等，最终使用空格将每个文档的词连接为字符串，实现文档的字符串表示。
    - 需要对每个文档进行分词、去除停用词、标点等处理。可以使用jieba分词库进行分词，使用NLTK库进行停用词的去除。在分词之前，需要先进行数据清洗，比如去除网址、表情符号等。处理完每个文档之后，将每个文档中的词语用空格连接起来，生成文档的字符串表示。

```
def word_process(sentence):
    # 使用jieba分词对文本进行分词操作，并利用停用词去除词语或标语等
    # words = [word for word in jieba.cut(sentence) if word.isalnum()]
    words = [line.strip() for line in
open('C:/Users/shiye/Desktop/python/cn_stopwords.txt', 'r', encoding='utf-
8').readlines()]  # 停用词表

    # 将分词结果使用空格连接成字符串
    result = ' '.join(words)
    return result
```

3. 文本的特征表示。实现一个模块，通过一个或多个函数，将每个文档转变为词频特征表示，以形成文档-词语的词频矩阵，可以选择使用sklearn中的CountVectorizer和TfidfVectorizer两种方式。
    - 需要将每个文档转化为词频特征表示，形成文档-词语的词频矩阵。可以使用sklearn中的CountVectorizer和TfidfVectorizer两种方式进行特征表示。需要注意的是，需要将处理后的文档列表转化为列表中的字符串表示。

```
if __name__ == '__main__':
    documents = ["节拍不对，也无所谓", "想唱就唱，想睡就睡", "你问快乐在哪里，快乐在这里。", "你问快乐在哪里，现在就告诉你"]
    docs = []
    for sentence in documents:
        docs.append(word_process(sentence))

    # 将文本转换成词频矩阵
    vectorizer = CountVectorizer()
    # 将文本转换成tf-idf矩阵

    X = vectorizer.fit_transform(docs)
```

4. 文本的话题分析。实现一个模块，通过一个或多个函数，借助sklearn.decomposition中的LatentDirichletAllocation构建主题模型（话题数目可以自主指定），并对发现的主题进行分析（每个主题对应的词语可利用model.components_来查看，每篇文档的主题概率分布可通过model.transform来查看），如利用f-string等进入输出。
    - 选择自主指定话题数目，利用model.components_查看每个主题对应的词语，利用model.transform查看每篇文档的主题概率分布，并进行输出。

```
    # 计算困惑度绘制elbow图确定主题数量
    perplexity_scores = []
    k_range = range(1, 6)  # 假设k的范围是1到5
    for k in k_range:
        lda = LatentDirichletAllocation(n_components=k)
        lda.fit(X)
        perplexity_scores.append(lda.perplexity(X))
    plt.plot(k_range, perplexity_scores, '-o')
    plt.xlabel('Number of topics')
    plt.ylabel('Perplexity')
    plt.show()

    k = 2

    # 使用LatentDirichletAllocation构建主题模型
    lda = LatentDirichletAllocation(n_components=k)
    lda.fit(X)

    # 输出每个主题对应的词语
    feature_names = vectorizer.get_feature_names_out()
    for i, topic in enumerate(lda.components_):
        print(f"Topic {i}:")
        top_words = [feature_names[j] for j in topic.argsort()[:-6:-1]]
        print(top_words)
```

5. 利用pickle或json对所得到的lda模型，对应的词频矩阵，以及特征表示，进行序列化保存，以备后面的pyLDAvis分析使用。
    - 使用pickle库对数据进行序列化，并将序列化后的结果保存到本地文件中。

```
    # 输出每篇文档的主题概率分布
    for i in range(len(documents)):
        print(f"Document {i}:")
        print(lda.transform(X[i]))
    # 输出结果

    pickle.dump((lda, X, vectorizer), open('./lda_model.pkl', 'wb'))
```

6. （附加）4中的超参数k（即话题的数目）变化时，评价LDA模型的一个指标，即困惑度（lda.perplexity），会随之波动 。观察不同k时困惑度的变化，并绘制曲线，看看有没有可能找到一个比较“优”的k。
7. （附加）考虑文档的时间，可否根据文档的话题分布来观察某话题随时间的变化趋势?
8. （附加）使用pyLDAvis对LDA模型进行可视化分析。

   参考资料：

   1. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html
   2. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html#sklearn.feature_extraction.text.TfidfVectorizer
   3. https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.LatentDirichletAllocation.html
   4. https://nbviewer.org/github/bmabey/pyLDAvis/blob/master/notebooks/pyLDAvis_overview.ipynb#topic=3&lambda=0.6&term=

完整代码：

```
import re
import jieba
import datetime
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.decomposition import LatentDirichletAllocation
import pickle

def sort_doc(file_path):
    # 定义时间段
    morning_start = datetime.time(4, 0, 0)
    noon_start = datetime.time(12, 0, 0)
    evening_start = datetime.time(20, 0, 0)

    # 定义数据分类字典
    data_by_time = \
        {
        "morning": [],
        "noon": [],
        "evening": []
        }
```

```
    # 读取数据文件
    with open(file_path, "r", encoding = "utf-8") as f:
        file_path:"C:/Users/shiye/Desktop/python/cn_stopwords.txt"
    for line in f:
        # 解析每一行数据
        fields = line.strip().split("\t")
        coords = fields[0]
        text = fields[1]
        time_str = fields[2]

        # 将时间字符串转换为datetime对象
        time = datetime.datetime.strptime(time_str, "%a %b %d %H:%M:%S %z
%Y").time()

        # 根据时间段将数据添加到相应的分类列表中
        if time >= morning_start and time < noon_start:
            data_by_time["morning"].append((coords, text))
        elif time >= noon_start and time < evening_start:
            data_by_time["noon"].append((coords, text))
        else:
            data_by_time["evening"].append((coords, text))
            return data_by_time

def word_process(sentence):
    # 使用jieba分词对文本进行分词操作，并利用停用词去除词语或标语等
    # words = [word for word in jieba.cut(sentence) if word.isalnum()]
    words = [line.strip() for line in
open('C:/Users/shiye/Desktop/python/cn_stopwords.txt', 'r', encoding='utf-
8').readlines()]  # 停用词表

    # 将分词结果使用空格连接成字符串
    result = ' '.join(words)
    return result

if __name__ == '__main__':
    documents = ["节拍不对，也无所谓", "想唱就唱，想睡就睡", "你问快乐在哪里，快乐在这
里。", "你问快乐在哪里，现在就告诉你"]
    docs = []
    for sentence in documents:
        docs.append(word_process(sentence))

    # 将文本转换成词频矩阵
    vectorizer = CountVectorizer()
    # 将文本转换成tf-idf矩阵

    X = vectorizer.fit_transform(docs)

    # 计算困惑度绘制elbow图确定主题数量
    perplexity_scores = []
    k_range = range(1, 6)  # 假设k的范围是1到5
    for k in k_range:
        lda = LatentDirichletAllocation(n_components=k)
        lda.fit(X)
        perplexity_scores.append(lda.perplexity(X))
    plt.plot(k_range, perplexity_scores, '-o')
```

```
    plt.xlabel('Number of topics')
    plt.ylabel('Perplexity')
    plt.show()

    k = 2

    # 使用LatentDirichletAllocation构建主题模型
    lda = LatentDirichletAllocation(n_components=k)
    lda.fit(X)

    # 输出每个主题对应的词语
    feature_names = vectorizer.get_feature_names_out()
    for i, topic in enumerate(lda.components_):
        print(f"Topic {i}:")
        top_words = [feature_names[j] for j in topic.argsort()[:-6:-1]]
        print(top_words)

    # 输出每篇文档的主题概率分布
    for i in range(len(documents)):
        print(f"Document {i}:")
        print(lda.transform(X[i]))
    # 输出结果

    pickle.dump((lda, X, vectorizer), open('./lda_model.pkl', 'wb'))
```

输出结果：