深度学习与自然语言处理第三次作业

李家哲 ZY2203105
lijiazhebuaa@buaa.edu.cn

作业要求

从给定的语料库中均匀抽取 200 个段落（每个段落大于 500 个词），每个段落的标签就是对应段落所属的小说。利用 LDA 模型对于文本建模，并把每个段落表示为主题分布后进行分类。验证与分析分类结果，（1）在不同数量的主题个数下分类性能的变化；（2）以"词"和以"字"为基本单元下分类结果有什么差异？

主要方法

M1: LDA 模型

LDA 是一种文档主题生成模型，也称为一个三层贝叶斯概率模型，包含词、主题和文档三层结构。

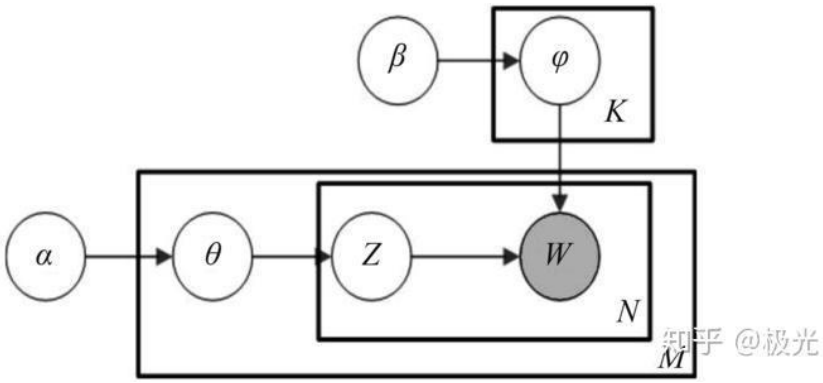

图 1　LDA 模型

作为一种非监督机器学习技术，可以用来识别大规模文档集或语料库中潜藏的主题信息。它采用了词袋的方法，这种方法将每一篇文档视为一个词频向量，从而将文本信息转化为了易于建模的数字信息。但是词袋方法没有考虑词与词之间的顺序，这简化了问题的复杂性，同时也为模型的改进提供了契机。每一篇文档代表了一些主题所构成的一个概率分布，而每一个主题又代表了很多单词所构成的一个概率分布。

举例来说，我们日常生活中总是产生大量的文本，如果每个文本存储为一篇文章，那么每篇文档从人的观察来说就是有序的词的序列 d=(w1，w1,...,wn)。

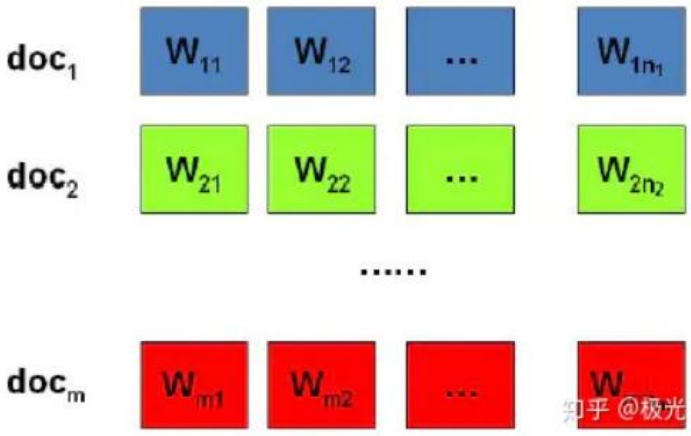

图 2　文档示意

包含 m 篇文章的语料库，每个文档有 Nm 个单词，一共涉及到 K 个主题；

每篇文档都有各自的主题，主题分布是多项式分布，该多项式分布的参数服从 Dirichlet 分布，该 Dirichlet 分布的参数为 α；

每个主题都有各自的词分布，词分布为为多项式分布，该多项式分布的参数服从 Dirichlet 分布，该 Dirichlet 分布的参数为 η；

对于某篇文档 d 中的第 n 个词，首先从该文档的主题分布中采用一个主题，然后再这个主题对应的词分布中采用一个词，不断重复该操作，直到 m 篇文档全部完成上述过程。

M2：随机森林（RF）

随机森林（Random Forest）是一种经典的 Bagging 模型，其弱学习器为决策树模型。如下图所示，随机森林模型会在原始数据集中随机抽样，构成 n 个不同的样本数据集，然后根据这些数据集搭建 n 个不同的决策树模型，最后根据这些决策树模型的平均值（针对回归模型）或者投票情况（针对分类模型）来获取最终结果。

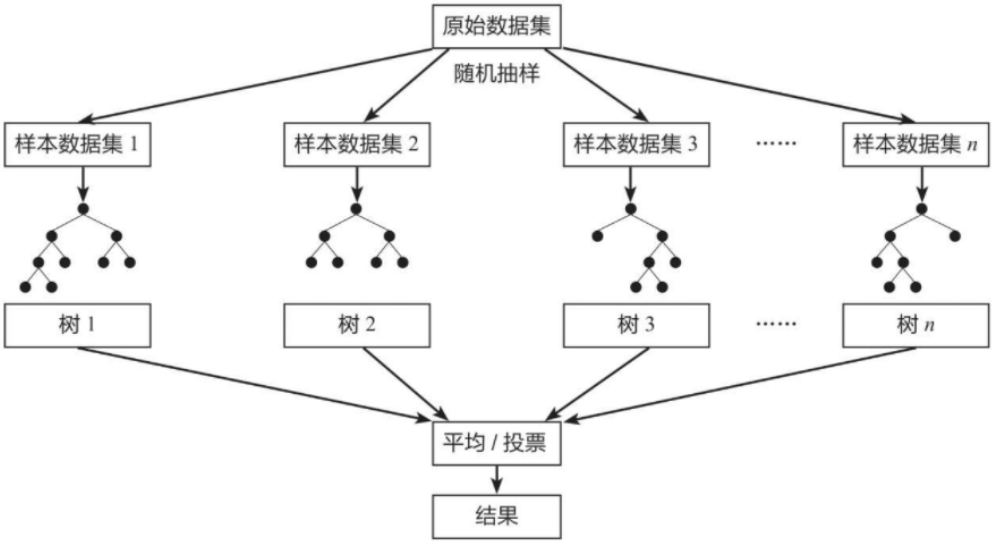

图 3　随机森林

M3：算法流程

step1：读取样本数据

选取金庸先生 10 本中长篇小说作为数据集，将每本小说分别按照“词”和“字”进行分词，并从每本小说中选取词数为 600 的 20 个段落，一共 200 个段落。

```
    def data_processing(file_path, flag, count, length):
    """
    获取文件信息并预处理
    :param file_path: 文件名对应路径
    :param flag: 选择词/字为单位，0=词，1=字
    :param count: 一本小说所分成的段落数
    :param length: 每个段落的词个数
    :return data_out: 字符串形式的语料库
    :return words_out: 分词
    :return paragraph_out: 随机选取的段落
    """
    # 读取小说
    delete_symbol = u'[a-zA-Z0-9’!"#$%&\'()*+,-./:: ;<=>?@，。?★、…【】《》？“”‘！[\\]^_`{|}~「」『』（）]+'
    with open(file_path, 'r', encoding='ANSI') as f:
        data_out = f.read()
        data_out = data_out.replace('本书来自 www.cr173.com 免费 txt 小说下载站\n 更多更新免费电子书请关注 www.cr173.com', '')
        data_out = re.sub(delete_symbol, '', data_out)
        data_out = data_out.replace('\n', '')
        data_out = data_out.replace('\u3000', '')
        data_out = data_out.replace(' ', '')
        f.close()
    # 读取并删除停词
    with open('./cn_stopwords.txt', 'r', encoding='utf-8') as f:
        stopwords = []
        for a in f:
            if a != '\n':
                stopwords.append(a.strip())
    for a in stopwords:
        data_out = data_out.replace(a, '')
    # 以字或词为单位进行分词
    words_out = []
    if flag == 0:
        words_out = list(jieba.cut(data_out))
    elif flag == 1:
        words_out = [c for c in data_out]
```

```
    # 将小说中的词分为 count 个段落
    paragraph_out = []
    for ii in range(count):
        begin = random.randint(0, len(words_out)-length-1)
        paragraph_out.append(words_out[begin:begin+length])

    return data_out, words_out, paragraph_out
```

---

step2：运行 LDA 算法和 RF 算法

分别测试主题数为 10 到 410 时分类的性能。

首先将每个段落表示为词袋模型，然后调用 gensim 库中的 LDA 模型进行训练，训练完成后将每个段落表示为主题分布，计算此时 LDA 模型的一致性得分和训练得到的主题中词频较高的 10 个特征词。

将 LDA 建模得到主题分布作为自变量，段落所属小说作为标签，以 8:2 的比例划分训练集和测试集，利用随机森林模型进行训练，在测试集上进行测试得到准确度。

```
    def lda_rf(paragraph_in, label_in, topics_in):
    """
    获取文件信
    :param paragraph_in: 200 个段落
    :param label_in: 200 个段落对应的标签
    :param topics_in: 主题数
    :return coherence_score_out: LDA 的一致性
    :return accuracy_out: 随机森林分类的准确度
    :return topic_words_out: 五个主题中前 10 个特征词
    """
    # 将每个段落表示为词袋模型
    dictionary = corpora.Dictionary(paragraph_in)
    corpus_bow = [dictionary.doc2bow(word) for word in paragraph_in]
    # 训练 LDA 模型
    lda_model = models.LdaModel(corpus_bow, num_topics=topics_in,
id2word=dictionary)
    # 将每个段落表示为主题分布
    corpus_lda = [lda_model[doc] for doc in corpus_bow]
    topic_probabilities = []
    for doc in corpus_lda:
        topic_temp = [0] * topics_in
        for topic, weight in doc:
            topic_temp[topic] = weight
        total_weight = sum(topic_temp)
        topic_probabilities.append([weight / total_weight for weight
in topic_temp])
```

```
    # 计算一致性得分
    cm = models.CoherenceModel(model=lda_model, texts=paragraph_in,
dictionary=dictionary, coherence='c_v')
    coherence_score_out = cm.get_coherence()
    # 取五个主题中前 10 个特征词
    topic_words_out = []
    for t in range(5):
        topic_word = lda_model.show_topic(t, topn=10)
        topic_words_out.append([word for word, _ in topic_word])

    # 划分训练集和测试集
    x = topic_probabilities
    y = label_in
    x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.2, random_state=42)

    # 训练随机森林
    rf = RandomForestClassifier()
    rf.fit(x_train, y_train)
    y_pred = rf.predict(x_test)
    # print(classification_report(y_test, y_pred))
    accuracy_out = metrics.accuracy_score(y_test, y_pred)

    return coherence_score_out, accuracy_out, topic_words_out
```

实验结果

按照上述算法流程进行实验，可以得到不同主题数下的 LDA 模型一致性结果如图 4 所示。LDA 模型的一致性得分越高，代表模型越好。

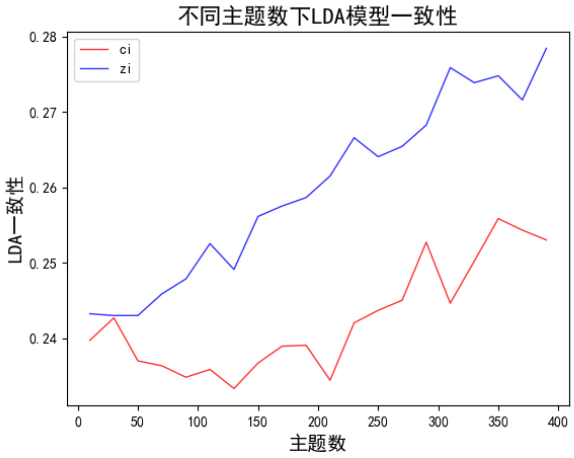

图 4　不同主题数下的 LDA 模型一致性得分

由图可以看出，随着主题数的增加，LDA 模型的一致性得分逐渐升高，代表主题分类越来越准确。因为选取的段落全部来自于金庸先生的武侠小说，题材是一样的，所以段落的特征词相差不多，在主题数较小的情况下，LDA 模型不够准确。并且以字为单位的模型训练一致性较高。

下表给出了其中五个主题中词频最高的前 10 个特征词。

表 1 以词为单位的特征词

	特征词
主题 0	道 听 陈家洛 见 便 中 说 说道 想 武功
主题 1	道 便 说 中 说道 见 甚 听 笑 武功
主题 2	道 便 中 说 说道 听 见 笑 武功 两
主题 3	道 说 便 说道 中 听 见 没 想 知道
主题 4	道 便 说 中 见 听 笑 说道 甚 袁承志

表 2 以字为单位的特征词

	特征词
主题 0	道 说 手 心 见 师 便 中 身 时
主题 1	道 子 说 中 见 手 身 心 出 想
主题 2	道 出 子 手 中 心 见 头 说 声
主题 3	道 中 说 见 出 三 心 子 声 剑
主题 4	道 说 中 子 心 见 师 时 出 知

可以看出主题中的特征词大多是与武侠相关的词，且都具有金庸先生小说的风格，如“武功”“陈家洛”等。但也有一些不好的情况，如“道”在每一个主题都有出现。

以 LDA 模型得到的主题分布进行分类训练，得到的准确度如图 5 所示。

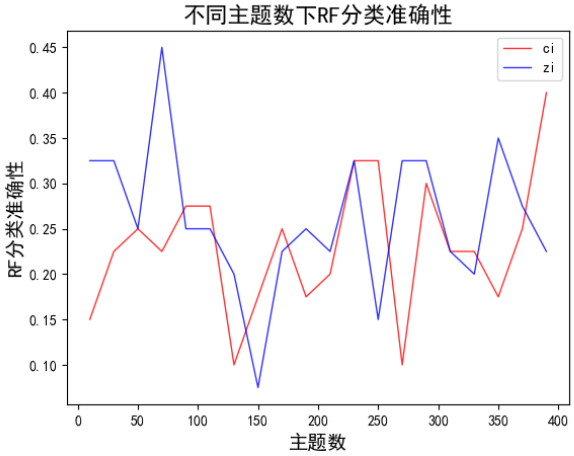

图 5 RF 分类准确度

可以看出以字和词为单位进行分类的结果相差不大。

结论

从以上结果可以得出结论：

（1）在不同主题数下，LDA 模型的一致性得分有较大的差异，随着主题数的增加，一

致性得分明显提高，模型建模效果明显提高。由于语料库中，每个段落都来自于武侠小说，风格较为统一，故需要较多的主题数才能获得满意的模型训练效果。从 RF 模型分类准确度可以看出，不同主题数下的准确度相差不太，且效果不太理想。

（2）以“词”和“字”为基本单元进行分类，其性能差异明显。从 LDA 模型一致性得分可以看出，以字为单位进行分类的一致性得分明显高于词；从 RF 分类准确度可以看出，以字和词为单位进行分类结果相差不大，总体来看以字为单位进行分类准确度较高于词。

参考文献

[1] 极光, LDA 模型, https://zhuanlan.zhihu.com/p/152594778

[2] 痴于代码, LDA 主题模型, https://blog.csdn.net/xiamaocheng/article/details/100599809

[3] 星幻夜极, LDA 随机森林模型, https://blog.csdn.net/m0_46388544/article/details/122758091