Report of Deep learning for Natural language Processing

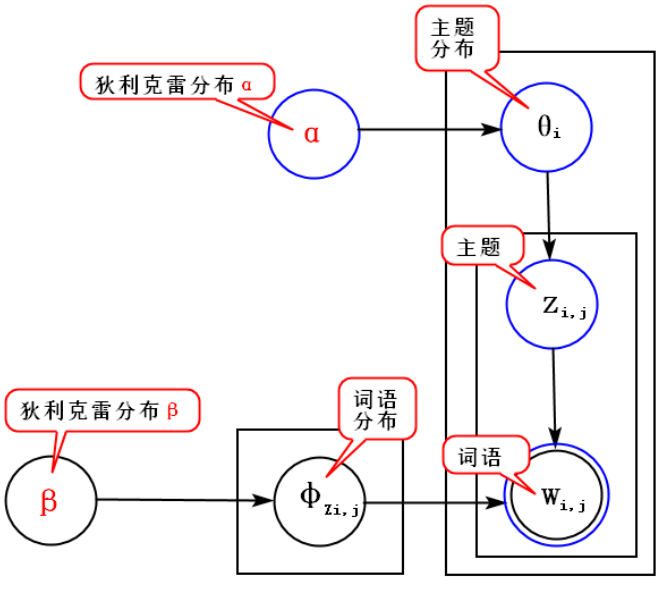
唐宗润 ZY2303211

Abstract

从语料库中均匀抽取 1000 个段落作为数据集（每个段落可以有 K 个 token, K 取 20，100，500, 1000, 3000）。利用 LDA 模型在给定的语料库上进行文本建模，主题数量为 T，并把每个段落表示为主题分布后进行分类（分类器自由选择），分类结果使用 10 次交叉验证（i.e. 900 做训练，剩余 100 做测试循环十次）。讨论在设定不同的主题个数 T 的情况下，分类性能的变化变化；以"词"和以"字"为基本单元下分类结果的差异；不同的取值的 K 的文本，主题模型性能上的差异。

Introduction

LDA（Latent Dirichlet Allocation）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。



所谓生成模型，就是说，我们认为一篇文章的每个词都是通过“文章以一定概率选择了某个主题，并从这个主题中以一定概率选择某个词语”这样一个过程得到。文档到主题服从多项式分布，主题到词服从多项式分布。LDA 是一种非监督机器学习技术,可以用来识别大规模文档集(document collection）或语料库（corpus）中潜藏的主题信息。

对于语料库中的每篇文档，LDA 定义了如下的生成过程：

1 对每一篇文档，从主题分布中抽取一个主题。

2 从上述被抽到的主题所对应的单词分布中抽取一个单词。

3 重复上述过程直至便利文档中的每一个单词。

LDA 认为每篇文章是由多个主题混合而成的，而每个主题可以由多个词的概率表征，所以整个程序的输入和输出如下所示：

类型	意义
输入	分词后的文章集（通常为一篇文章一行),主题数K，超参数α 和 β
输出	1、每篇文章的各个词被置顶的主题编号 2、每篇文章的主题概率分布 3、每个主题下的词概率分布. 4、程序中词语word的id映射表. 5、每个主题下从高到低topn特征词

Methodology

首先对数据进行预处理，将文件中停词去掉，广告去除。

```python
def read_cn_stopwords():
    cn_stopwords = []
    with open('D:/zongruntang/nlp/second/jyxstxtqj_downcc.com/cn_stopwords.txt', mode='r', encoding='utf-8') as f:
        cn_stopwords.extend([line.strip() for line in f.readlines()])
    return cn_stopwords


1个用法
def read_novel(path):
    para_list = []
    para_label = []
    # 读取小说列表
    with open(file=path + 'inf.txt', mode='r', encoding='gb18030') as f:
        file_names = f.readline().split(',')
    f.close()
    # 读取小说 去掉广告 去掉较短的句子 剩下的段落文本放list 生成对应的索引放label
    for index, name in enumerate(file_names):
        novel_name = path + name + '.txt'
        with open(file=novel_name, mode='r', encoding='gb18030') as f:
            content = f.read()
            ad = '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'
            content = content.replace(ad, _new: '')
            for sentence in content.split('\n'):
                if len(sentence) < 500:
                    continue
                para_list.append(sentence)
                para_label.append(index)

    return para_list, para_label
```

其次，对预处理后的段随机抽取 1000 个，书名为标签

```python
def para_extract(para, label):
    # 段落抽取 随机选择1000个段落，后面函数进行token处理
    text_ls = []
    text_label = []
    random_indices = random.sample(range(len(para)), k: 1000)
    text_ls.extend([para[i] for i in random_indices])
    text_label.extend([label[i] for i in random_indices])
    return text_ls, text_label
```

对选中段落进行分词，利用 jieba 分词，再此基础上加个循环实现以字为基本单元的数组

```python
def split(text_ls, text_label):
    # 分词
    stop_words = read_cn_stopwords()
    tokens_word = []  # 以词为单位
    tokens_word_label = []
    tokens_char = []  # 以字为单位
    tokens_char_label = []
    for i, text in enumerate(text_ls):
        words = [word for word in jieba.lcut(sentence=text) if word not in stop_words]
        tokens_word.append(words)
        tokens_word_label.append(text_label[i])

        temp = []
        for word in words:
            temp.extend([char for char in word])
        tokens_char.append(temp)
        tokens_char_label.append(text_label[i])
```

构建字典和向量集后，进行 LDA 模型训练。

```python
1个用法
def train_lda_model(documents, num_topics):

    dictionary = corpora.Dictionary(documents)
    corpus = [dictionary.doc2bow(doc) for doc in documents]

    print('Training LDA model')
    # lda_model = gensim.models.LdaModel(corpus, num_topics=num_topics, id2word=dictionary, passes=20)
    lda_model = gensim.models.LdaMulticore(corpus, num_topics=num_topics, id2word=dictionary, passes=20, workers=4)
    print('Finished training LDA model')

    return lda_model,dictionary,corpus
```

获得主题分布

```python
#返回的结果就是一个二维数组，其中每一行表示一个文档的主题分布
1个用法
def get_document_topic_distribution(lda_model, documents):
    topic_distributions = []
    for doc in documents:
        bow_vector = lda_model.id2word.doc2bow(doc)
        topic_distribution = lda_model.get_document_topics(bow_vector, minimum_probability=0.0)
        topic_distribution = [score for _, score in topic_distribution]
        topic_distributions.append(topic_distribution)
    return np.array(topic_distributions)
```

划分测试集和数据集，并进行交叉验证

```python
#利用交叉验证给分类器打分
1个用法
def evaluate_classification_performance(X, y, classifier):
    kf = KFold(n_splits=10, shuffle=True, random_state=42)
    scores = cross_val_score(classifier, X, y, cv=kf)
    return scores.mean()
```

Conclusion

以字为单位和以词为单位的 LDA 模型测试集准确度如下。

	T	T=20	T=40	T=60	T=80
char	k=20	0.133	0.148	0.139	0.133
	k=100	0.174	0.176	0.2	0.172
	k=500	0.291	0.265	0.231	0.248
	k=1000	0.269	0.308	0.314	0.306
	k=3000	0.248	0.244	0.278	0.256
word	k=20	0.271	0.248	0.242	0.212
	k=100	0.41	0.342	0.404	0.373
	k=500	0.724	0.696	0.707	0.663
	k=1000	0.662	0.729	0.722	0.688
	k=3000	0.709	0.735	0.76	0.732

（1） 在设定不同的主题个数 T 的情况下，分类性能是否有变化?

当 K 值固定时，T 在【20，60】内，准确度随着 T 值增大而增大，而当 T 在【60，80】时，准确度反而下降，说明 Topics 数量过多，可能导致分类过于细化而影响准确度

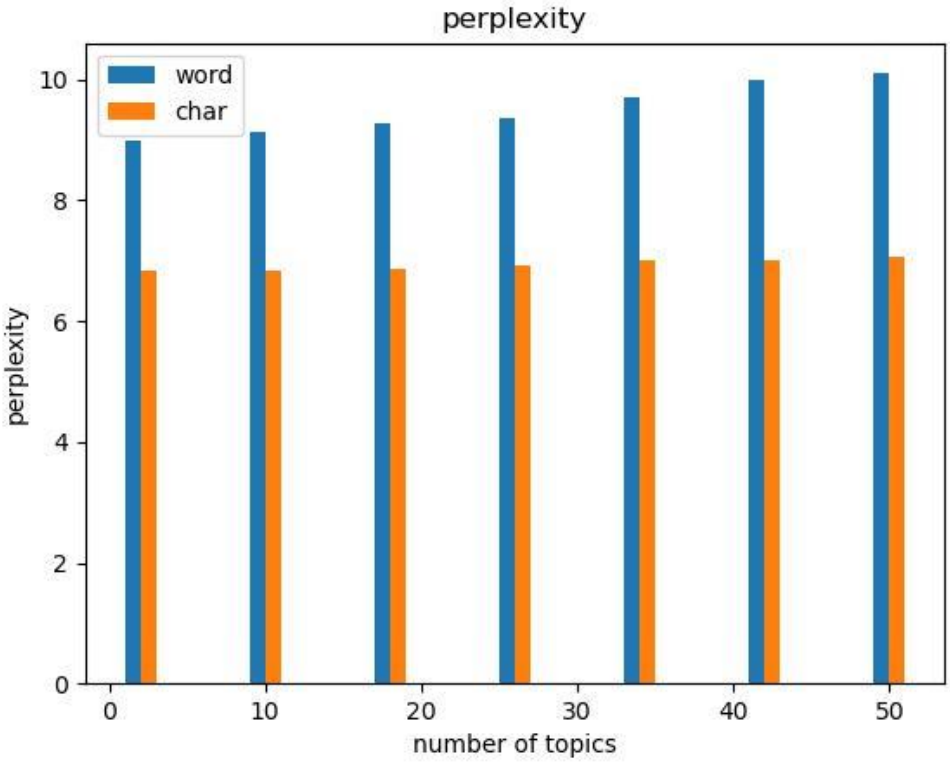
（2） 以"词"和以"字"为基本单元下分类结果有什么差异?

当 T 和 K 过小时，两者差距不大，当 T 和 K 取适当值时（如 T=40，K=3000），以字为基本单位的准确度低于以词为基本单位的准确度。

（3） 不同的取值的 K 的短文本和长文本，主题模型性能上是否有差异?

当 T 值固定时（如取 T=40）,以字为基本单位或者以词为基本单位，k 值越大，取得的 token 数越多，准确度越高

最后，为了评价 LDA 模型的性能，引入困惑度评价指标，计算如图


perplexity
word
char
perplexity
0
2
4
6
8
10
0
10
20
30
40
50
number of topics