时空数据分析与挖掘

——文本挖掘与主题分析

学　院：遥感信息工程学院

班　级：2006 班（20F10）

姓　名：马文卓

学　号：2020302131249

老　师：田扬戈

时　间：2023 年 4 月 30 日

目录

一、实验目标与内容

1. 实验目标

根据文本挖掘和主题分析的相关知识，对所给数据中的 Article Title（可选）、Author Keywords、Keywords Plus、Abstract（可选）和其他字段进行文本挖掘和主题分析，找到数据挖掘领域的主要研究方向，并简单描述；你觉得哪些方向比较有前途，给出解释。

2. 数据与要求

（1）数据

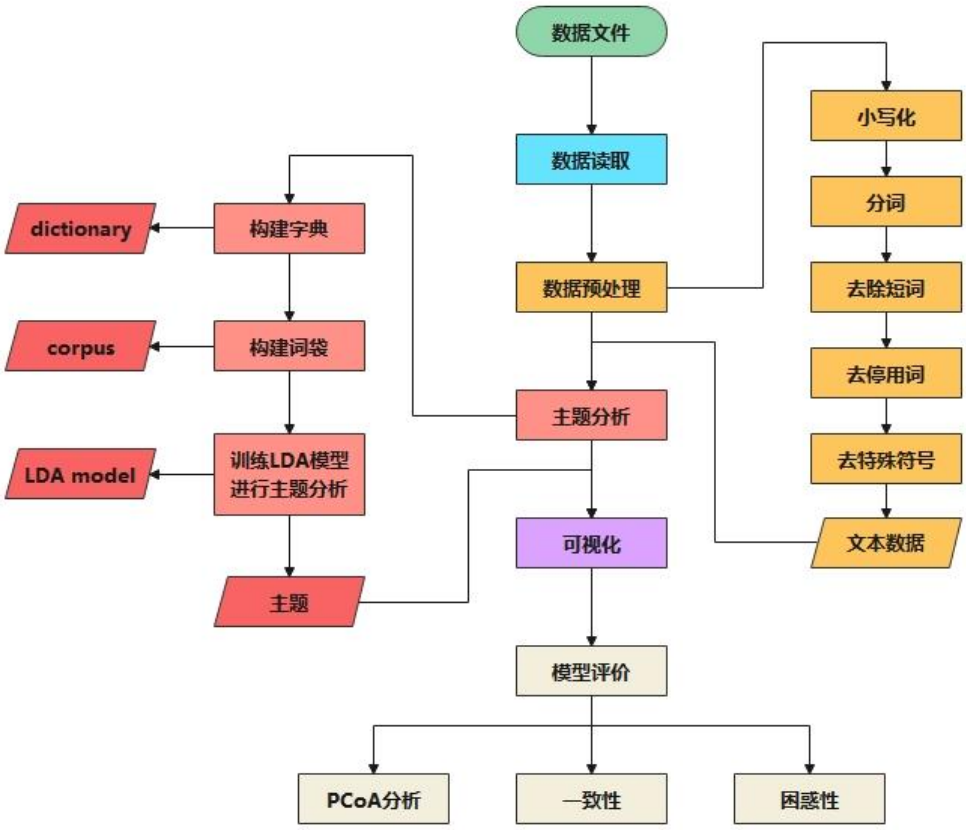
本次实验的数据为：SCI 数据库近 5 年所有有关 data mining 的论文的全部记录。数据一共 16071 篇论文记录，每个论文记录包含了 72 个字段。本次实验我们主要使用到其中的【Author Keywords】和【Keywords Plus】字段，对所有论文的关键词进行主题的挖掘与分析。

（2）要求

①请大家根据 Article Title（可选）、Author Keywords、Keywords Plus、Abstract（可选）和其他字段，进行文本挖掘和主题分析
②找到数据挖掘领域的主要研究方向
③对这些主题进行合理的解释分析
④提供代码和简单报告
⑤报告里应有代码运行结果和分析

二、具体实施步骤与源码

1. 步骤流程图



本次实验中，主要流程为上图所示。主要步骤分为数据读取、数据预处理、主题分析、可视化、模型评价五个步骤（详细步骤解析和源码见本章第三小节）。具体而言，数据预处理部分主要包含小写化、分词、去除短词、去停用词、去特殊符号等操作。主题分析步骤主要包含构建数据字典、构建词袋、训练 LDA 模型进行主题分析等操作。可视化主要包含主题输出、词云绘制等操作。模型评价部分主要包含主坐标分析（PCoA）、一致性分析、困惑性分析等。在上述步骤完成之后，还进行了一系列的分析实验，以进一步的完善实验，分析实验部分可见第三章。

2. 代码结构图

- 语言：Python
- 版本：3.7.16
- 第三方库：gensim、nltk、wordcloud、pyLDAvis、pandas、os、re、pickle

**具体代码可见 code 目录*

```python
if not os.path.exists(dictionary_path) or not os.path.exists(corpus_path) or not os.path.exists(lda_path): # 如果之前数据不存在
    print('之前不存在数据,需生成...')
    # 数据读取与预处理
    database=readFile(data_folder,['Author Keywords','Keywords Plus','Publication Year']) # 读关键词
    database=dataPrepoccess(database,numPaperInDoc) # 处理关键词字段，得到关键词库
    dictionary=corpora.Dictionary(database) # 生成字典
    dictionary.save_as_text(dictionary_path) # 保存字典
    # 生成词袋bag of word
    corpus=[dictionary.doc2bow(text) for text in database]
    with open(corpus_path, 'wb') as f:
        pickle.dump(corpus, f) # 保存corpus
    # 使用 LDA 模型进行主题分析
    lda_model = models.LdaModel(
        corpus, # 词袋
        num_topics=num_topics, # 挖掘出的主题个数
        id2word=dictionary, # 字典
        passes=10) # 训练过程中穿过语料库的次数
    lda_model.save(lda_path) # 保存模型
else: # 之前保存过数据…

    # 保存主题挖掘结果
    with open(topic_path, 'w') as f:
        # 循环写入每个主题下的词语
        for topic in lda_model.print_topics():
            f.write(str(topic) + '\n')
    # 可视化
    createWordCloud(lda_model,image_path) # 绘制每个主题的词云
    # 分析
    vis_pcoa = pyLDAvis.gensim_models.prepare(lda_model, corpus, dictionary, sort_topics=False)# 主坐标PCoA分析
    pyLDAvis.save_html(vis_pcoa, pcoa_path) # 存为html文件
    return lda_model,dictionary,corpus,database
```

数据读取
数据预处理

主题挖掘

可视化

模型评价

3. 具体实施步骤与源码解析

（1）数据读取

从给定的数据文件夹中读取所有的 Excel 数据文件中的指定字段。本次实验当中我们读取 Author Keywords、Keywords Plus 和 Publication Year 这三个字段，来进行主题的挖掘。其中读文件使用 pandas 的 read_excel 方法，注意这里做了空缺值的处理，并且最后按照发布年份进行了从小到大的排序，这样做的目的是为了后续实验（按照发布年份归为一个文档的实验）的方便。

```python
def readFile(path,constraint=None,no_values=''):
    '''
    读取数据文件,返回list
    params:
        path: 数据文件夹路径
        constraint: 读取指定列的列名称列表
        no_values: 空缺值处理
    returns:
        data_sum:数据列表
    '''
    files=os.listdir(path)
    data_sum=[]
    for filename in files:
        file=os.path.join(path,filename)
        data=pd.read_excel(io=file,usecols=constraint,engine='xlrd') # 读取文件
        data.fillna(no_values,inplace=True) # 填写空缺值
        data=data.values.tolist() # 转为list
        data_sum+=data
    data_sum.sort(key=lambda x:str(x[2])) # 按发表年份排序
    return data_sum
```

```python
# 数据读取与预处理
database=readFile(data_folder,['Author Keywords','Keywords Plus','Publication Year']) # 读关键词
```

（2）数据预处理

数据预处理中包含了小写化、分词、去短词、去停用词、去特殊符号等操作，经过这些操作之后的数据列表还可以通过手工剔除的方式进行进一步的清洗，以去除掉一些代码没能剔除干净且无关紧要的关键词。

如下图为数据预处理的函数，值得注意的是 numPaperInDoc 参数，其代表一个文档由几篇文章构成，其设置的目的是为了方便做后续的对比实验。

```
def dataPrepoccess(data,numPaperInDoc):
    '''
    数据预处理,组成database
    params:
        data:处理数据
        numPaperInDoc:由几篇文章构成一个文档
    returns:
        database:处理后的数据
    '''
    database=[]
    stopwords=['data mining']
    keys_cleaned=[]
    for i in range(len(data)):···
    return database
```

①小写化：为了关键词的一致统一，我们将所有字母转化为小写。

```
# 全部小写化
data[i][0]=data[i][0].lower()
data[i][1]=data[i][1].lower()
```

②分词：以’;’为分割符，将文本中的关键词分割出来。

```
# 分割关键词
a=data[i][0].split('; ') # 分割Author Keywords字段
b=data[i][1].split('; ') # 分割Keywords Plus字段
keys=a+b
```

③去短词：由于根据实际情况，长度为 1 和 2 的英文单词基本没有什么实际含义，而长度为 3 的英文单词大多为一些缩写（如 knn 等），所以在本次实验中我将长度小于 3 的关键词进行剔除。

```
if len(key)<3: # 删除长度太短的字符
    continue
```

④去停用词：由于本次实验的数据位近五年来的与数据挖掘相关的论文，所以很明显最后挖掘出来的主题关键词中一定有 data mining，且是占比很大的，为了不影响我们挖掘更有意义的主题，这里我将‘data mining’和‘’作为停用词进行剔除。

```
if key in stopwords: # 删除停用词
    continue
```

⑤去除特殊符号：由于关键词列表中还有很多带有特殊符号的无意义词汇（如$#等），所以我们要对特殊符号进行有选择的剔除。值得注意的是，’-’和’,’不能剔除，因为’-’一般在关键词中起连词的作用（如 k-means），’,’一般起或者的作用如（2,3d）。所以这里我使用正则匹配替换掉除去’-’和’,’的其他特殊字符。

```
key=re.sub(r'[^A-Za-z0-9,\s\-]+','',key) # 替换特殊字符
```

⑥文档构建：将每 numPaperInDoc 篇文章中的所有经过处理的关键词构建成一个文档。注意如果 numPaperInDoc=0 者按照 Publication Year 字段构建文档，即同年发表的文章构建一个文档。

```
if numPaperInDoc==0: # 同年文章构成一个文档
    if i!=len(data)-1 and data[i][2]!=data[i+1][2]: # 下个paper的发布年份和现在这个paper不一致
        database.append(keys_cleaned) # 纳入数据库
        keys_cleaned=[]
else:
    if (i+1)%numPaperInDoc==0: # 每numPaperInDoc篇文章构成一个document
        database.append(keys_cleaned) # 纳入数据库
        keys_cleaned=[]
```

（3）主题挖掘

LDA（Latent Dirichlet Allocation）模型是一种常用的主题模型，它可以发现文本背后的潜在主题。在 LDA 模型中，文档是由多个主题组成的，每个主题是由多个词语组成的。LDA 模型的目标是从一组文档中发现潜在主题，并为每个文档和主题分配概率。

本次实验中使用 gensim 库来创建 LDA 模型进行主题挖掘和分析。其主要步骤包含构建数据字典、构建词袋、训练 LDA 模型进行主题分析。详细如下：

①构建数据字典：首先使用 corpora.Dictionary 类构造数据字典，其原理是遍历原始数据列表，给每个词分配一个唯一的整数 id，并且统计其出现的次数，构成一个字典。最后将字典保存为 txt 文件，以备后续使用。

```
dictionary=corpora.Dictionary(database) # 生成字典
dictionary.save_as_text(dictionary_path) # 保存字典
```

②构建词袋：使用 doc2bow 方法将字典中的关键词转化为词袋。其主要过程为遍历了 texts 中的每一个文本，使用 dictionary.doc2bow()方法将文本转化为了一个稀疏向量。这个稀疏向量中包含了文本中所有出现过的词汇以及对应的出现次数，其中对应的词汇通过在 dictionary 中的映射得到。最后，它将所有的稀疏向量组成了一个列表，也就是我们常说的词袋。最后将词袋保存为 pkl 文件，以备后续使用。

```
# 生成词袋bag of word
corpus=[dictionary.doc2bow(text) for text in database]
with open(corpus_path, 'wb') as f:
        pickle.dump(corpus, f) # 保存corpus
```

③训练 LDA 模型：LDA 模型的训练过程是通过 Gibbs 采样算法来实现的。具体步骤如下：

1>初始化每个单词的主题：对于每个文档中的每个单词，随机指定一个主题。这里的主题是指 LDA 模型中的主题，而不是一般意义上的主题。

2>对于每个单词 w_i，计算其在每个主题 k 下的条件概率 $P(z=k|w_i)$，即单词 w_i 属于主题 k 的概率。这个概率可以根据下面的公式计算：

$$p(z=k|w_i)=\frac{n_{k,w_i}+\beta}{\sum_w n_{k,w}+V\beta}\times\frac{n_{d,k}+\alpha}{\sum_{k'} n_{d,k'}+K\alpha}$$

其中，n_{k,w_i} 表示主题 k 中包含单词 w_i 的数量，$n_{d,k}$ 表示文档 d 中属于主题 k 的单词数量，V 表示字典中不同单词的数量，K 表示主题的数量，ß和α是两个超参数。

3>对于每个单词 w_i，根据其在不同主题下的条件概率，随机选择一个主题，将其指定为该单词的主题。

4>重复步骤 2 和步骤 3，直到收敛为止。这里的收敛指的是模型参数稳定，或者达到指定的迭代次数。

5>根据估计出的主题和单词的分布，计算每个主题下的关键词，并输出结果。

具体的构建训练代码如下，最后保存模型以备后续使用。

```
# 使用 LDA 模型进行主题分析
lda_model = models.LdaModel(
    corpus, # 词袋
    num_topics=num_topics, # 挖掘出的主题个数
    id2word=dictionary, # 字典
    passes=10) # 训练过程中穿过语料库的次数
lda_model.save(lda_path) # 保存模型
```

（4）可视化

本次实验中的可视化模块主要包含主题的输出保存和词云的绘制。

①主题的保存输出：之前训练好的 LDA 模型挖掘出的主题输出保存到 txt 文件中。

```
# 保存主题挖掘结果
with open(topic_path, 'w') as f:
    # 循环写入每个主题下的词语
    for topic in lda_model.print_topics():
        f.write(str(topic) + '\n')
```

②词云的绘制：使用 wordcloud 库将每个主题中的主要关键词以词云的形式可视化出来。

```
def createWordCloud(lda_model,image_path):
    '''
    绘制词云
    params:
        lda_model: 训练好的lda模型
        image_path: 词云图片保存路径
    '''
    fig, axs = plt.subplots(ncols=2, nrows=math.ceil(lda_model.num_topics/2), figsize=(16,20))
    axs = axs.flatten()

    def color_func(word, font_size, position, orientation, random_state, font_path):
        return 'darkturquoise'

    for i, t in enumerate(range(lda_model.num_topics)):

        x = dict(lda_model.show_topic(t, 30))
        im = WordCloud(
            background_color='black',
            color_func=color_func,
            max_words=4000,
            width=300, height=300,
            random_state=0
        ).generate_from_frequencies(x)
        axs[i].imshow(im.recolor(colormap= 'Paired_r' , random_state=244), alpha=0.98)
        axs[i].axis('off')
        axs[i].set_title('Topic '+str(t))
    # vis
    plt.tight_layout()
    plt.savefig(image_path)# 保存图片
```

（5）模型评价

本次实验中我们主要使用主坐标分析、一致性和连贯性等指标来分析评价模型的质量。

①主坐标分析 PCoA：这里使用欧几里得距离作为度量，也就是 PCoA 分析编程 PCA 分析，将之前所得到的的主题降维至 2 个分量。我们通过 pyLDAvis 库来进行这个 PCoA 的分析和可视化。

```
# 分析
vis_pcoa = pyLDAvis.gensim_models.prepare(lda_model, corpus, dictionary, sort_topics=False)# 主坐标PCoA分析
pyLDAvis.save_html(vis_pcoa, pcoa_path) # 存为html文件
return lda_model,dictionary,corpus,database
```

②一致性和困惑性评价：一致性和困惑性被广泛用作主题模型的评估指标。

一致性是对主题质量的度量，它指示提取的主题是否易于人类理解。计算方法是使用学习语料库使用对数条件概率来找到每个主题的词间相似度的平均值，并且如果整个主题的连贯性很高，则被认为是一个很好的模型。

困惑性是指模型对未见过的新数据的预测能力，也称为困惑度。它通常用于评估主题模型的泛化能力，即模型对新数据的适应能力。困惑性越低，表示模型对新数据的预测能力越好，模型的质量也越好。

本次实验中使用 UMass coherence 和困惑性来评价模型好坏，以寻找一个合适的主题挖掘数量（num of topics）。将主题数量由 2 增加到 20，对于每个主题数量都进行主题挖掘（训练一个 LDA 模型），并且计算该模型的一致性和困惑性，最后绘制成图。

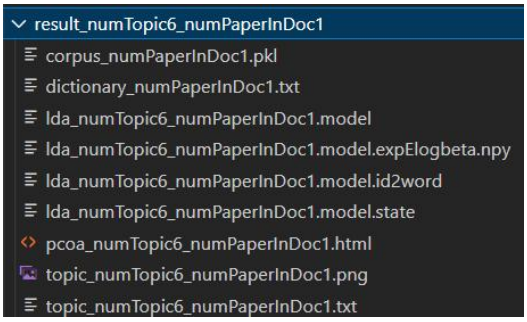
```
coherence_vals = [] # 一致性
perplexity_vals = [] # 困惑性

for n_topic in range(start,stop,step):
    lda_model,dictionary,corpus,texts = TextMining(data_folder='/home/ubuntu/mwz/Spatio-temporal_data_mining_and_analysis/data'
                result_folder=f'/home/ubuntu/mwz/Spatio-temporal_data_mining_and_analysis/Text-Mining/result/result_numTopic{n_t
                num_topics=n_topic,
                numPaperInDoc=numPaperInDoc) # 主题挖掘
    perplexity_vals.append(np.exp2(-lda_model.log_perplexity(corpus))) # 计算困惑性
    coherence_model_lda = CoherenceModel(model=lda_model, texts=texts, dictionary=dictionary, coherence='u_mass') # 计算一致性
    coherence_vals.append(coherence_model_lda.get_coherence())
```

```
# 绘图
x = range(start, stop, step) # 横坐标
fig, ax1 = plt.subplots(figsize=(12,5))
# 一致性
c1 = 'darkturquoise'
ax1.plot(x, coherence_vals, 'o-', color=c1)
ax1.set_xlabel('Num Topics')
ax1.set_ylabel('Coherence', color=c1); ax1.tick_params('y', colors=c1)
# 困惑性
c2 = 'slategray'
ax2 = ax1.twinx()
ax2.plot(x, perplexity_vals, 'o-', color=c2)
ax2.set_ylabel('Perplexity', color=c2); ax2.tick_params('y', colors=c2)
# 可视化
ax1.set_xticks(x)
fig.tight_layout()
plt.savefig(f'/home/ubuntu/mwz/Spatio-temporal_data_mining_and_analysis/Text-Mining/analysis/analysis_numT
```

三、实验结果分析

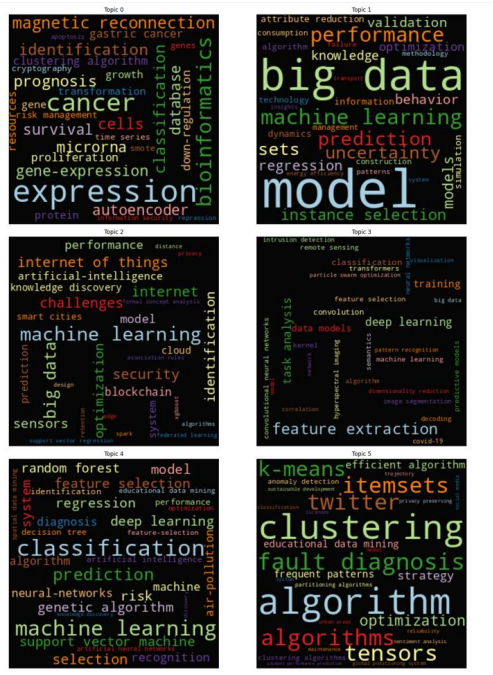
本次实验我做了多组对比实验，以找到最好的挖掘结果。值得注意的是，我主要在两个参数上进行研究对比，一个参数是挖掘主题的数目（numTopic）另外一个是每个文档构成的 paper 数量（numPaperInDoc）。如下图所示，对于每个参数组合，其挖掘结果都存储在 result 目录下的对应文件夹当中，每个结果文件夹中包含以下 9 个文件：

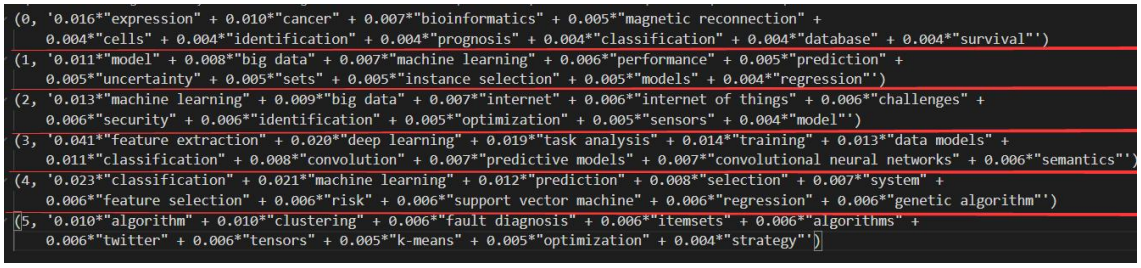
1>corpus_numTopic_numPaperInDoc.pkl：词袋变量
2>dictionary_numTopic_numPaperInDoc.txt：数据字典文本
3>lda_numTopic_numPaperInDoc.model(.npy,.id2word,.state)：LDA 模型
4>pcoa_numTopic_numPaperInDoc.html：主题 PCoA 分析结果网页
5>topic_numTopic_numPaperInDoc.png：主题词云图
6>topic_numTopic_numPaperInDoc.txt：主题输出文本



1. 主题分析

本次实验最终我以 numTopic=6 且 numPaperInDoc=1 的 LDA 模型作为最终的模型（选择这个的原因在后续的分析中讲解，本小节主要针对该模型说挖掘的主题进行分析），该模型挖掘的主题结果如下：





从上述的结果来看，一共挖掘得到 6 个主题，可进行如下分析。

（1）TOPIC ONE

主题一中关键词按照占比由大到小排列为：expression、cancer、bioinformatics、magnetic reconnection、cells、identification、prognosis、classification、database、survival。

可以非常明显的判断出 TOPIC ONE 为医疗主题。因为其关键词中的 cancer、bioinformatics、cells、prognosis、survival 等都是与医疗领域相关的关键词。其实观察这五年来 data mining 的发展趋势，在各行各业的运用越来越广泛，挖掘出这个主题并不意外。随着大数据时代的到来，传统医疗越来越跟不上时代的步伐。有句话说的好，互联网行业，有些人负责开发游戏提供娱乐消遣、有些人负责设计应用便捷社会，但是也必须要有人负责——改变时代！而大数据时代下，数据挖掘就是这个 era-changer。对于医疗行业而言，在线医疗就是大数据时代的医疗创新，在这当中，data mining 作为数据的整合者、分析者，可以将海量的医疗资源整合到一起，然后通过文本挖掘、主题分析、分类、学习等多种手段挖掘其中的有用信息，以提供给医生更加准确的支持并给予病人更加便捷的医疗环境。

所以关于数据挖掘的文章中得到医疗相关的主题并不奇怪，而且我坚信，海量数据结合数据挖掘为基础的医疗产业才是互联网时代下的新趋势。

（2）TOPIC TWO

主题二中关键词按照占比由大到小排列为：model、big data、machine learning、performance、prediction、uncertainty、sets、instance selection、models、regression。

TOPIC TWO 为大数据主题。由 big data、model、machine learning、sets、uncertainty 等关键词，我更倾向于将这个 topic 归为大数据主题。由于数据挖掘的发展根本是大数据时代的来临，换句话说正是因为现在的数据量太大才让数据挖掘有了存在的意义。因此，数据挖掘对于大数据领域的研究只会更加频繁，只有了解到海量数据的更多信息和规律，才能更好的促进数据挖掘领域的发展和突破。

（3）TOPIC THREE

主题三中关键词按照占比由大到小排列为：machine learning、big data、internet、internet of things、challenges、security、identification、optimization、sensors、model。

TOPIC THREE 为应用主题。通过该主题可以得知 data mining 的应用领域涵盖了大数据、机器学习、互联网、物联网、安全、识别、优化、传感器等等。其实如此广泛的应用范围只是数据挖掘强大能力的一个缩影，数据挖掘旨在通过海量数据挖掘出有用的信息，因此对于任何需要以海量数据为基础的应用领域都应该是数据挖掘大展拳脚的舞台，我认为数据挖掘会在大数据时代的任何领域展露

锋芒。

（4）TOPIC FOUR

主题四中关键词按照占比由大到小排列为：feature extraction、deep learning、task analysis、training、data models、classification、convolution、predictive models、convolutional neural networks、semantics。

TOPIC FOUR 为**深度学习主题**。通过 deep learning、training、convolution、convolution neural networks 等关键词我更倾向于认为这个 topic 主要是阐述 DL 相关主题。近几年来深度学习飞速席卷所有领域，在大部分领域替代了传统方法。DL 的核心是模型从大量数据当中学习数据的分布、特征等，这其实与数据挖掘的要义（从大量数据当中获取有用的信息）一致。因此，数据挖掘和深度学习本身就很难单独分离开来，两者的融合贯通已然是必然。总体而言，深度学习其实就是一种数据挖掘技术，而传统的数据挖掘算法也能够给现在深度学习网络提供有用的信息，因此而言 data mining 和 deep learning 的发展前景是极其乐观的。

（5）TOPIC FIVE

主题五中关键词按照占比由大到小排列为：classification、machine learning、prediction、selection、system、feature selection、risk、support vector machine、regression、genetic algorithm。

TOPIC FIVE 为**传统机器学习主题**。通过 machine learning、classification、support vector machine、regression 等关键词我更倾向于认为这个 topic 主要是阐述 ML 相关主题。传统机器学习是根基，其中的分类、回归、聚类等算法更是现有很多算法的核心。由于 ML 也是基于大量数据的，数据挖掘依然依赖于传统的机器学习算法，包括一些人工智能算法（GA）。因此，数据挖掘领域对于机器学习的研究是很有必要的，在现在数据挖掘技术的支持下，一些传统的机器学习算法也许能更新换代，得到新时代的智能机器学习算法。这个研究方向固然是令人神往的，但是鉴于传统机器学习已经达到瓶颈，说个人认为研究前景没有特别好。

（6）TOPIC SIX

主题六中关键词按照占比由大到小排列为：algorithm、clustering、fault diagnosis、itemsets、algorithms、twitter、tensors、k-means、optimization、strategy。

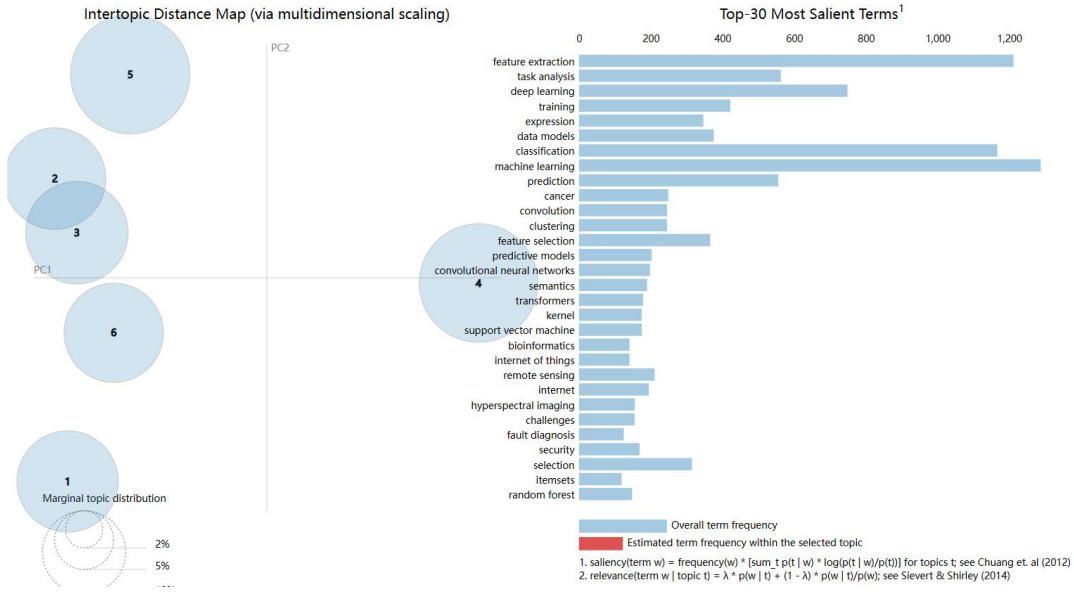
TOPIC SIX 为**算法主题**。通过 algorithm、clustering、k-means、optimization 等关键词我倾向于将此 topic 归为算法主题。和前面两个主题类似，数据挖掘给很多的传统算法提供了无限的可能，因此对于这些算法的研究是很有必要切有前途的。其实我们观察发现，在多个主题中多次出现的，除去 big data、ML 等之外就是 classification，也就是分类算法。所以可见，数据挖掘领域对于分类算法的研究是非常多的，而且也映射出现实社会对于分类这一任务的迫切需求。

2. 模型评价

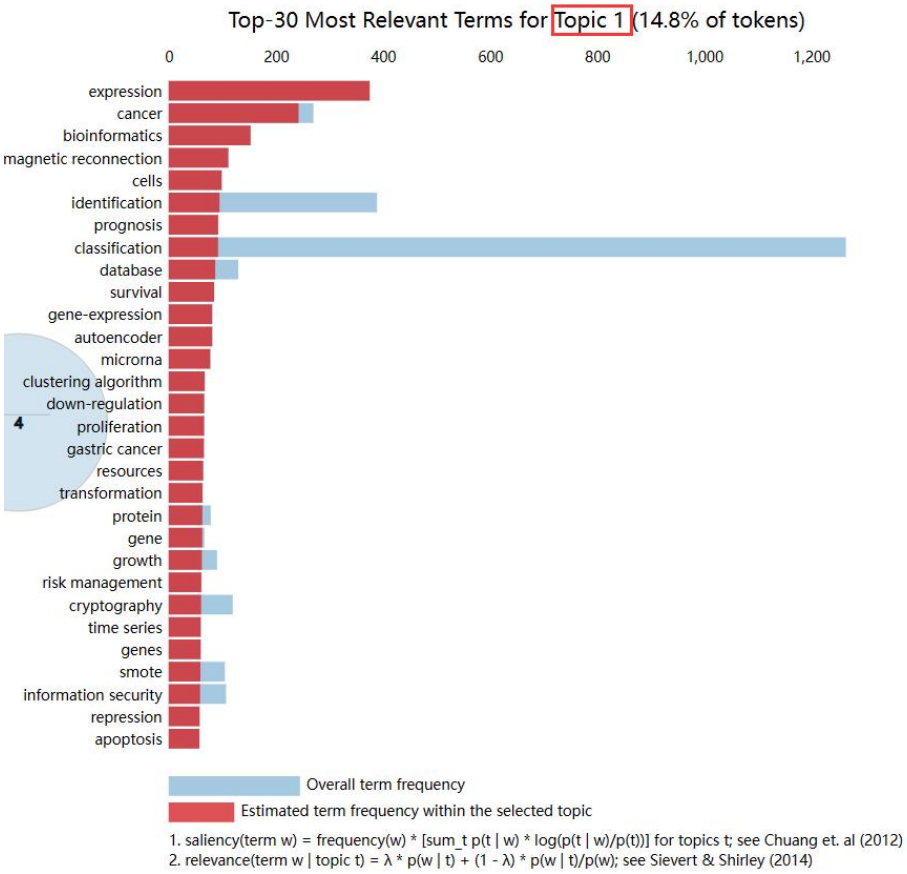
对上述模型（numTopic=6 且 numPaperInDoc=1）挖掘出的主题进行了 PCoA 分析（这里使用欧几里得距离，所以其实就是 PCA 分析）。得到结果如下所示（详细可见对应的 html 文件）。

我们可以看到的是这六个主题重合部分很少（只有 topic2 和 topic3 有少部分重合）。其中 topic4（深度学习主题）最为特殊，其次是 topic1（医疗主题），

再其次是 topic5（机器学习主题），说明这三个主题是提炼的比较好的。

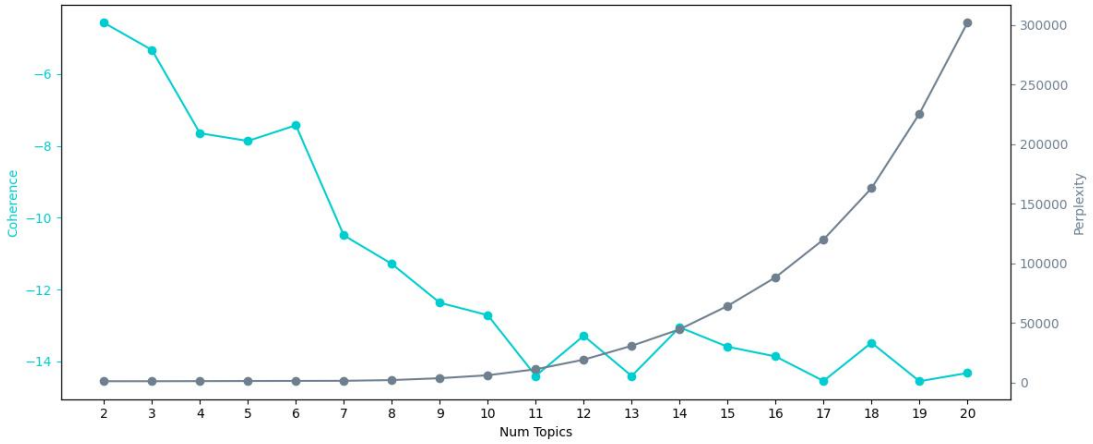


我们单独看 topic1（医疗主题）。可以看到，基本所有的 cancer、bioinformatics、prognosis、survival、microrna、proliferation、gene 等于医疗相关的词在所有的 tokens 中占比都很大（很多 100%）。



3. 主题数量分析

由于挖掘主题的数量不同也会影响结果的质量，因此我做了这样的对比实验：将主题数量 numTopic 从 2 增加到 20，对于每一个主题数量都进行主题挖掘，得到 LDA 模型，并且计算其一致性和困惑性，最后绘制成图。根据一致性（较高的好）、困惑性（较低的好）综合选择主题数量。值得一提的是，这样的实验我们针对不同的 numPaperInDoc（每个文档中 paper 的数量）都进行了，这里我们只展示分析 numPaperInDoc=1 的组别（其他组别可见 analysis 目录）。



如上图所示，为不同主题数量模型对应的一致性和困惑性绘制成图。模型一致性越高说明主题之间的区分度越高，模型质量越好；困惑性越低说明模型的对待新数据的泛化能力越好，模型质量越好。由于主题数的增加，主题之间肯定相似度越来越高，也很难泛化，所以一致性整体呈下降趋势、困惑性整体呈上升趋势。

单从这两个指标来看，应该选择 numTopic=2 的模型作为最好的模型（一致性最高、困惑性最低）。但是考虑到主题的多样性，这里我们选择 numTopic=6 作为最优模型。

4. 文档关键词数量分析

由于每个文档的关键词数量不同也会导致不同的挖掘质量，所以这里我针对每个主题数量的模型都做了三种不同的文档划分方式的实验。

1>第一种 numPaperInDoc=1，即每篇 paper 的关键词组成一个 document，比较具备实际意义，但是每个 document 的词比较少。

2>第二种 numPaperInDoc=10，即每 10 篇 paper 的关键词组成一个 document，没什么实际意义，但是每个 document 的词比较合理。

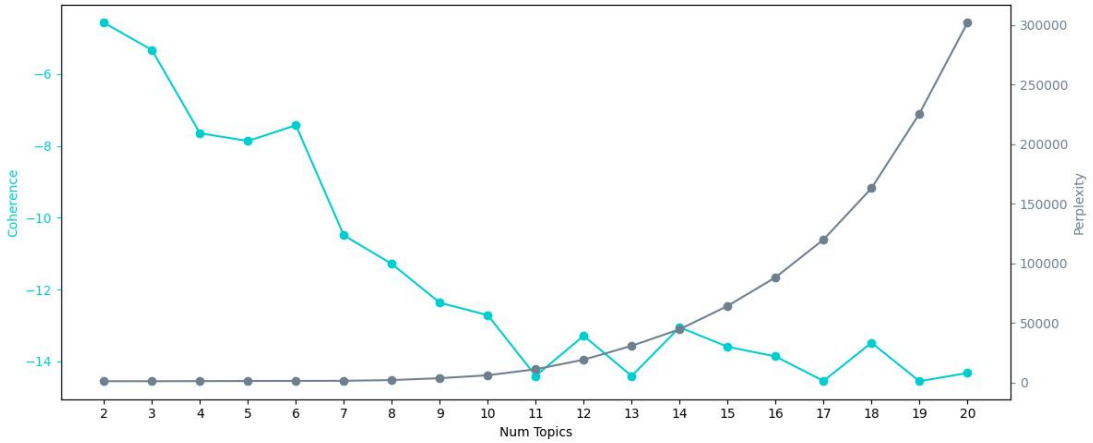
3>第三种 numPaperInDoc=0，即按照每篇 paper 的 Publication Year 将原始文本分为 5 个文档，分别为‘没有发表年份’、‘2019’、‘2020’、‘2021’、‘2022’。

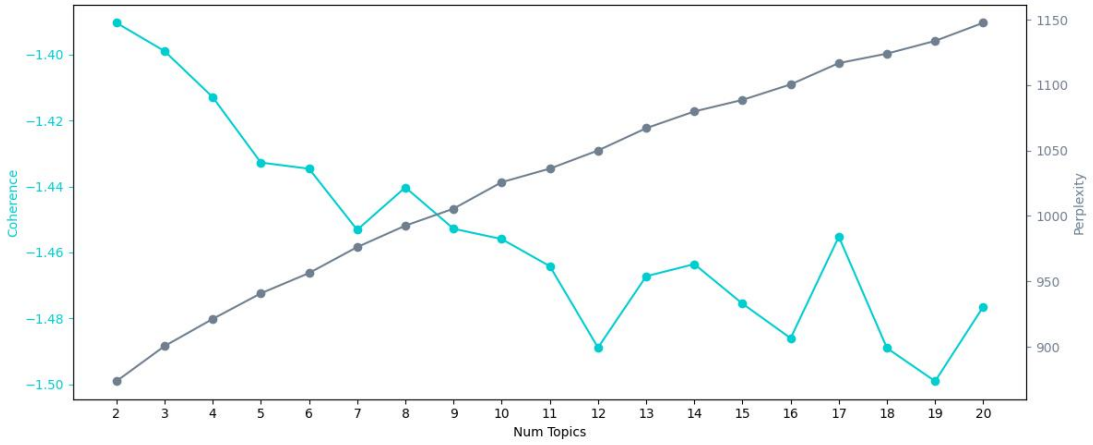
得到的实验结果如下图所示。其实我们可以看到，文档中关键词数量最为合理的（图 2），其一致性是最高的，其实按照指标而言应该选择 numPaperInDoc=10。但是考虑到实际意义，毕竟每篇文章代表不同作者的思考，所以我仍然固执的选择 numPaperInDoc=1 作为结果进行展示。

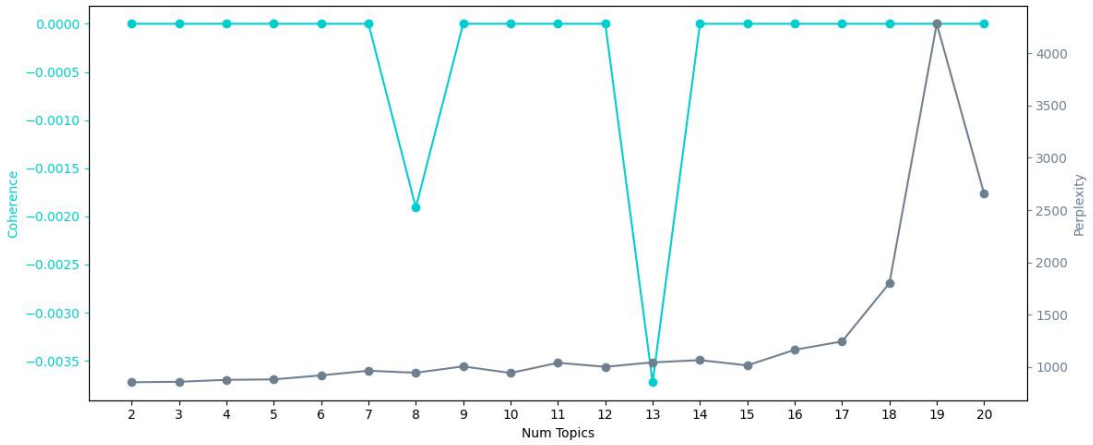
其实通过词云和 PCoA 分析等手段，我们会发现，当一个文档中包含的 paper 数量太多之后（例如图三，按照发布年份进行归档，每个文档当中都有数以万计的关键词），会导致虽然计算的一致性很高（趋近于 0），但是其实他只是包含

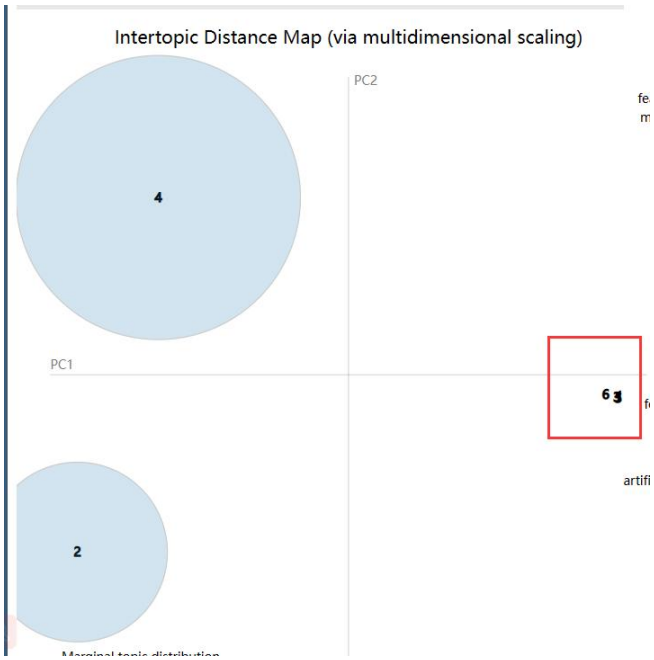
大部分词语的主题之间区分很大，很多少量词主城的主题之间是趋同的（图四），所以这样也是不可取的。

（其实是可以选择更好的指标对应的模型，但是本人更像挖掘出的主题更加具有实际意义，所以选择了 numPaperInDoc=1 的模型作为展示，当然指标更好的模型结果在 result 目录下都有记录）









四、附录

在本次实验之前我考虑使用 Article Title 和 Abstract 字段。具体的操作方法是，对这两个字段进行分词、数据清理等，然后和关键词字段合并在一起，然后进行主题分析，这样就解决了关键词较少的问题。

于是按照上述想法我做了实验（代码可见 code 目录下 text_mining_try.py），得到如下结果。

我们可以看到挖掘出来的主题当中有很多没有意义的词汇，如 Proposed、data、mining、method、chinese、used 等等。所以我认为这个结果是不好的。再者考虑到 Author Keywords 和 Keywords Plus 两个字段就是作者根据文章内容浓缩的主题，所以在上述实验当中我主要使用两个关键词字段（和 Publication Year 字段）。