文本挖掘从小白到精通（四）---文本相似度检索

AINLP 3天前

以下文章来源于Social Listening与文本挖掘 ， 作者Scottish Fold Cats

Social Listening与文本挖掘

爱好文本分析，社会化媒体数据挖掘

写在前面：笔者最近在梳理自己的文本挖掘知识结构，借助gensim、sklearn、keras等库的文档做了些扩充，会陆陆续续介绍文本向量化、tfidf、主题模型、word2vec，既会涉及理论，也会有详细的代码和案例进行讲解，希望在梳理自身知识体系的同时也能对想学习文本挖掘的朋友有一点帮助，这是笔者写该系列的初衷。

前面一篇文章聊到了各类主题模型，这些主题模型除了可以发现语料中的潜在主题外，还可以用于抽取新增文档的特征，用于后续的文本分类、文本聚类等任务。本文首先来聊聊基于LSI的文本相似度检索（Text Similarity Queries）。

正式开始前，设置日志和工作环境，培养码代码的好习惯，打印程序运行中的细节，以便后面找到报错。

```
import os
import tempfile
from pprint import pprint
import logging
logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s', level=l
TEMP_FOLDER = tempfile.gettempdir()
print('文件夹"{}" 将被用来存储语料和临时性的字典'.format(TEMP_FOLDER))
```

文件夹"C:\Users\hp\AppData\Local\Temp" 将被用来存储语料和临时性的字典

一、简易的相似度查询接口（Simple Similarity Interface）

在之前两篇文章--- 语料库和词向量空间 和 主题模型和文本数据转换 中，笔者介绍了在向量空间模型（Vector Space Model ）中创建语料库（Corpus）意味着什么，以及如何进行各类不同的文本向量转换。

现在，我们将使用之前做的LSI模型，来实现一个实际任务 --- 相似文档检索。相似文档检索，或者文档相似度检测，在日常生活中有很多实际的应用：

- 论文抄袭检测
- 搜索引擎
- 内容推荐
- ...

基本原理：

> 我们需要通过文本数据转换来实现文档对的相似度（Similarity between Pairs of Documents）查询，或给定一个待查询文档，发现该文档和其他一组文档的相似度（Similarity between a Specific Document and a Set of Other Documents），比如，用户会在某个文档检索系统（比如论文查询系统）中输入一个检索词汇/语句，然后系统会按文本相似度的降序排列，从上到下依次显示和检索条件最为相似的文档。

为了说明在gensim中如何做到这一点的，笔者将使用一个新的语料库进行演示，这些语料来自笔者之前发布在**人人都是产品经理**专栏上的文章标题集合（专栏链接：http://www.woshipm.com/u/107483），感兴趣的可以根据链接进行查阅。

```
from gensim import corpora, models, similarities
from collections import defaultdict
from pprint import pprint  #使打印的格式更齐整
import jieba
```

```
#对特定长词进行控制，防止被分错词，影响后续的分析效果
jieba.add_word('微信')
jieba.add_word('文本挖掘')
jieba.add_word('增长黑客')
jieba.add_word('小白')
jieba.add_word('大数据')

```

```
docs = [
'数据挖掘实操丨用文本挖掘剖析近5万首《全唐诗》',
'以虎嗅网4W+文章的文本挖掘为例，展现数据分析的一整套流程',
'干货丨作为一个合格的“增长黑客”，你还得重视外部数据的分析！',
'文本挖掘从小白到精通（二）---语料库和词向量空间',
'文本挖掘从小白到精通（三）---主题模型和文本数据转换',
'文本挖掘从小白到精通（一）---语料、向量空间和模型的概念',
'以《大秦帝国之崛起》为例，来谈大数据舆情分析和文本挖掘',
'文本分类算法集锦，从小白到大牛，附代码注释和训练语料',
'Social Listening和传统市场调研的关系是怎样的？',
'【新媒体运营实操】如何做出一个精美的个性化词云'
'以哈尔滨冰雪大世界旅游的传播效应为例，谈数据新闻可视化的“魅惑”',
'万字干货丨10款数据分析“工具”，助你成为新媒体运营领域的“增长黑客”',
'如何用数据分析，搞定新媒体运营的定位和内容初始化？',
'当数据分析遭遇心理动力学：用户深层次的情感需求浮出水面',
'揭开微博转发传播的规律：以“人民日报”发布的G20文艺晚会微博为例',
'数据运营实操 | 如何运用数据分析对某个试运营项目进行“无死角”的复盘？',
'如何利用微信后台数据优化微信运营',
'如何利用Social Listening从社会化媒体中“提炼”有价值的信息？',
'用大数据文本挖掘，来洞察“共享单车”的行业现状及走势',
'从社交媒体传播和文本挖掘角度解读《欢乐颂2》',
'不懂数理和编程，如何运用免费的大数据工具获得行业洞察？',
'写给迷茫的你：如何运用运营思维规划自己的职业发展路径？',
'如何用聚类分析进行企业公众号的内容优化',
'傅园慧和她的“洪荒之力”的大数据舆情分析',
'数据运营|数据分析中，文本分析远比数值型分析重要！（上）'
        ]
```

再对文本进行**分词**，用**空格隔开**组成字符串，方便进行下一步的处理：

```
documents = [' '.join(jieba.lcut(i)) for i in docs]
documents
```

['数据挖掘 实操 | 用 文本挖掘 剖析 近 5 万首 《 全唐诗 》',
'以虎 嗅网 4W + 文章 的 文本挖掘 为例 ， 展现 数据分析 的 一整套 流程',
'干货 | 作为 一个 合格 的 “ 增长黑客 ” ， 你 还 得 重视 外部 数据 的 分析 ！ ',
'文本挖掘 从小 白到 精通 （ 二 ） - - - 语料库 和 词 向量 空间',
'文本挖掘 从小 白到 精通 （ 三 ） - - - 主题 模型 和 文本 数据 转换',
'文本挖掘 从小 白到 精通 （ 一 ） - - - 语料 、 向量 空间 和 模型 的 概念',
'以 《 大秦 帝国 之 崛起 》 为例 ， 来谈 大数据 舆情 分析 和 文本挖掘',

'文本 分类 算法 集锦， 从小 白到 大牛， 附 代码 注释 和 训练 语料',
'Social Listening 和 传统 市场调研 的 关系 是 怎样 的？',
'【 新 媒体 运营 实操 】 如何 做出 一个 精美 的 个性化 词云以 哈尔滨 冰雪 大 世界 旅游 的 传播效应 为 例， 谈 数据 新闻 可视化 的“ 魅惑 ”',
'万字 干货 | 10 款 数据分析“ 工具 ”， 助 你 成为 新 媒体 运营 领域 的“ 增长黑客 ”',
'如何 用 数据分析， 搞定 新 媒体 运营 的 定位 和 内容 初始化？',
'当 数据分析 遭遇 心理 动力学： 用户 深层次 的 情感 需求 浮出 水面',
'揭开 微博 转发 传播 的 规律： 以“ 人民日报 ”发布 的 G20 文艺晚会 微博为 例',
'数据 运营 实操 | 如何 运用 数据分析 对 某个 试运营 项目 进行“ 无 死角 ”的 复盘？',
'如何 利用 微信 后台 数据 优化 微信 运营',
'如何 利用 Social Listening 从 社会化 媒体 中“ 提炼 ”有 价值 的 信息？',
'用 大数据 文本挖掘， 来 洞察“ 共享 单车 ”的 行业 现状及 走势',
'从 社交 媒体 传播 和 文本挖掘 角度 解读 《 欢乐颂 2 》',
'不 懂 数理 和 编程， 如何 运用 免费 的 大数据 工具 获得 行业 洞察？',
'写给 迷茫 的 你： 如何 运用 运营 思维 规划 自己 的 职业 发展 路径？',
'如何 用 聚类分析 进行 企业 公众 号 的 内容 优化',
'傅园慧 和 她 的“ 洪荒 之力 ”的 大数据 舆情 分析',
'数据 运营 | 数据分析 中， 文本 分析 远比 数值 型 分析 重要！ （ 上 ）']

```
# 去停用词
stoplist = [i.strip() for i in open('datasets/stopwords_zh.txt',encoding='utf-8
texts = [[word for word in document.lower().split() if word not in stoplist]
for document in documents]

pprint(texts)
```

[['数据挖掘', '实操', '文本挖掘', '剖析', '万首', '全唐诗'],
['以虎', '嗅网', '4w', '文章', '文本挖掘', '为例', '展现', '数据分析', '一整套', '流程'],
['干货', '合格', '增长黑客', '重视', '外部', '数据', '分析'],
['文本挖掘', '白到', '精通', '语料库', '向量', '空间'],
['文本挖掘', '白到', '精通', '主题', '模型', '文本', '数据', '转换'],
['文本挖掘', '白到', '精通', '语料', '向量', '空间', '模型', '概念'],
['大秦', '帝国', '崛起', '为例', '来谈', '大数据', '舆情', '分析', '文本挖掘'],
['文本', '分类', '算法', '集锦', '白到', '大牛', '代码', '注释', '训练', '语料'],
['social', 'listening', '传统', '市场调研', '关系'],
['媒体',
'运营',
'实操',
'精美',
'个性化',
'词云以',
'哈尔滨',
'冰雪',
'旅游',

'传播效应',
'为例',
'数据',
'新闻',
'可视化',
'魅惑'],
['万字', '干货', '数据分析', '工具', '媒体', '运营', '领域', '增长黑客'],
['数据分析', '搞定', '媒体', '运营', '定位', '内容', '初始化'],
['数据分析', '遭遇', '心理', '动力学', '用户', '深层次', '情感', '需求', '浮出', '水面'],
['揭开', '微博', '传播', '规律', '人民日报', '发布', 'g20', '文艺晚会', '微博为'],
['数据', '运营', '实操', '数据分析', '试运营', '项目', '死角', '复盘'],
['微信', '后台', '数据', '优化', '微信', '运营'],
['social', 'listening', '社会化', '媒体', '提炼', '价值', '信息'],
['大数据', '文本挖掘', '洞察', '共享', '单车', '行业', '现状及', '走势'],
['社交', '媒体', '传播', '文本挖掘', '角度', '解读', '欢乐颂'],
['数理', '编程', '免费', '大数据', '工具', '行业', '洞察'],
['写给', '迷茫', '运营', '思维', '规划', '职业', '路径'],
['聚类分析', '企业', '公众', '内容', '优化'],
['傅园慧', '洪荒', '之力', '大数据', '舆情', '分析'],
['数据', '运营', '数据分析', '文本', '分析', '远比', '数值', '分析']]

下面再用词袋模型（Bag-of-Words）来提取文本特征，对该操作不太了解的同学，可以参看前面的文章：

```
dictionary = corpora.Dictionary(texts)
corpus = [dictionary.doc2bow(text) for text in texts]
```

2019-05-13 12:47:05,520 : INFO : adding document #0 to Dictionary(0 unique tokens: [])
2019-05-13 12:47:05,524 : INFO : built Dictionary(126 unique tokens: ['万首', '全唐诗', '剖析', '实操', '数据挖掘']...) from 24 documents (total 187 corpus positions)

得到的词袋表示是这样的，没有出现在dictionary里的词汇将不会显示。

```
# 查看词袋表示中的部分数据
list(corpus)[:3]
```

[[(0, 1), (1, 1), (2, 1), (3, 1), (4, 1), (5, 1)],
[(5, 1),
(6, 1),
(7, 1),
(8, 1),
(9, 1),
(10, 1),
(11, 1),
(12, 1),

```
 (13, 1),
 (14, 1)],
[(15, 1), (16, 1), (17, 1), (18, 1), (19, 1), (20, 1), (21, 1)]]
```

基于上面构建好的词袋表示语料库，笔者用LSI模型构建一个10维（只设定10个主题）的LSI向量空间（10-Dimensional LSI Space）。

在这里，因为参与训练的文档数量不多，笔者暂且将主题数设置为10个，如果语料足够的多，主题数理论上也要设定得够多，因为主题数也代表文本数据中的特征数量，捕捉到的特征越多，就越能提升模型的效果；

该模型中另一个值得注意的一个参数是power_iters，模型中的默认设为2，设置的数值越大，模型的精确度就越高，尤其是在训练语料较少的情况下，但训练语料过多的话，加大该数值会影响模型的运行效率。

```
lsi = models.LsiModel(
          corpus,
          id2word=dictionary,
          power_iters=100,
          num_topics=10
          )
```

```
2019-05-13 12:50:21,182 : INFO : using serial LSI version on this node
2019-05-13 12:50:21,186 : INFO : updating model with new documents
2019-05-13 12:50:21,186 : INFO : preparing a new chunk of documents
2019-05-13 12:50:21,188 : INFO : using 100 extra samples and 100 power iterations
2019-05-13 12:50:21,189 : INFO : 1st phase: constructing (126, 110) action matrix
2019-05-13 12:50:21,191 : INFO : orthonormalizing (126, 110) action matrix
2019-05-13 12:50:21,838 : INFO : 2nd phase: running dense svd on (110, 24) matrix
2019-05-13 12:50:21,844 : INFO : computing the final decomposition
2019-05-13 12:50:21,845 : INFO : keeping 10 factors (discarding 35.084% of energy spectrum)
2019-05-13 12:50:21,846 : INFO : processed documents up to #24
2019-05-13 12:50:21,847 : INFO : topic #0(4.917): 0.414*"运营" + 0.377*"数据" + 0.314*"数据分析" + 0.271*"文本挖掘" + 0.264*"分析" + 0.243*"媒体" + 0.188*"为例" + 0.186*"实操" + 0.151*"文本" + 0.131*"白到"
2019-05-13 12:50:21,848 : INFO : topic #1(4.227): -0.499*"文本挖掘" + -0.356*"白到" + -0.295*"精通" + 0.275*"运营" + -0.208*"向量" + -0.208*"空间" + -0.202*"模型" + -0.176*"语料" + -0.143*"大数据" + 0.140*"媒体"
2019-05-13 12:50:21,850 : INFO : topic #2(3.632): -0.399*"大数据" + -0.380*"分析" + 0.256*"白到" + -0.227*"舆情" + 0.178*"精通" + -0.173*"洞察" + -0.173*"行业" + 0.150*"语料" + -0.130*"数据分析" + 0.130*"模型"
2019-05-13 12:50:21,852 : INFO : topic #3(3.520): 0.285*"分析" + 0.250*"文本" + -0.246*"为例" + -0.227*"文本挖掘" + 0.223*"数据分析" + -0.213*"媒体" + -0.167*"实操" + -0.157*"冰雪" + -0.157*"可视化" + -0.157*"传播效应"
2019-05-13 12:50:21,853 : INFO : topic #4(3.376): 0.417*"数据分析" + -0.296*"分析" + -0.235*"数据" + 0.174*"心理" + 0.174*"浮出" + 0.174*"需求" + 0.174*"情感" + 0.174*"用户" + 0.174*"动力学" + 0.174*"遭遇"
```

现在，笔者设定的检索语句为“**文本挖掘在舆情口碑挖掘中的作用很大**”（需做分词处理）， 笔者想让结果按相似度的降序进行排列展示，也就是和查询语句相似度越高（相似度数值较大）的展示在上面。

与我们经常使用的搜索引擎（如百度搜索、Google搜索）不同，在这里我们只是单纯的考虑文本（词汇）的**显式语义相关性**。没有考虑超链接的数量、随机游走静态排名等因素，只是对布尔关键字匹配（Boolean Keyword Match）的语义扩展：

```
#查询语句为“文本挖掘在舆情口碑挖掘中的作用很大”
doc = "文本挖掘 在 舆情 口碑 挖掘 中 的 作用 很大 "
vec_bow = dictionary.doc2bow(doc.lower().split())
vec_lsi = lsi[vec_bow]       #将查询语句转换到LSI向量空间


result = [(docs[i[0]],i[1]) for i in vec_lsi]
pprint(sorted(result,key=lambda x: x[1],reverse=True))
```

[('数据挖掘实操 | 用文本挖掘剖析近5万首《全唐诗》', 0.33455623240887034),
('以《大秦帝国之崛起》为例，来谈大数据舆情分析和文本挖掘', 0.13033674738237558),
('文本分类算法集锦，从小白到大牛，附代码注释和训练语料', 0.0779247512458276),
('文本挖掘从小白到精通（三）---主题模型和文本数据转换', 0.024538387740754494),
('【新媒体运营实操】如何做出一个精美的个性化词云以哈尔滨冰雪大世界旅游的传播效应为例，谈数据新闻可视化的“魅惑”',
-0.082419463003527),
('Social Listening和传统市场调研的关系是怎样的？', -0.08250316554091632),
('文本挖掘从小白到精通（一）---语料、向量空间和模型的概念', -0.10281448121396503),
('文本挖掘从小白到精通（二）---语料库和词向量空间', -0.257795594292202),
('干货 | 作为一个合格的“增长黑客”，你还得重视外部数据的分析！', -0.31021073457393633),
('以虎嗅网4W+文章的文本挖掘为例，展现数据分析的一整套流程', -0.5668413112759387)]

从上面的结果看来，基本符合文本相似度检索的要求，但精度不够，还有提升的空间。

因此，笔者将考虑用余弦相似性（Cosine Similarity）来度量两个向量之间的相似性。

> **余弦相似性通过测量两个向量的夹角的余弦值来度量它们之间的相似性。0度角的余弦值是1，而其他任何角度的余弦值都不大于1；并且其最小值是-1。从而两个向量之间的角度的余弦值确定两个向量是否大致指向相同的方向。两个向量有相同的指向时，余弦相似度的值为1；两个向量夹角为90°时，余弦相似度的值为0；两个向量指向完全相反的方向时，余弦相似度的值为-1。这结果是与向量的长度无关的，仅仅与向量的指向方向相关。余弦相似度通常用于正空间，因此给出的值为0到1之间。余弦相似性最常见的应用就是计算文本的相似度。**
>
> **考虑最简单的情况，有2个文档，通过词袋表示、TF-IDF或者word2vec等模型将这两个文档表示为2个向量，计算这两个向量的余弦值，就可以知道两个文本在统计学方法中他们的相似度情况。实践证明，这是一个非常有效的方法。**

余弦相似度是向量空间建模中的标准度量，但有一种情况是不可以的 --- 当向量表示概率分布时（这在主题模型中很常见），这需要使用其他的相似性度量来完成，比如Jellinger、Kullback_leibler等距离度量方

式，笔者将会在后面的文章中提及。

二、初始化查询结构（Initializing Query Structures）

在相似性查询的准备阶段，我们需要输入后面参与检索的文档，在这里，笔者还是使用上面的语料库，但在实际的应用场景中，参与检索的文档和之前用于训练模型的文档通常不是同一批。

```
index = similarities.MatrixSimilarity(lsi[corpus])  #将查询语料库转换到LSI向量空间
#内存友好型接口
#index = similarities.Similarity(output_prefix='Similarity',corpus=lsi[corpus],
```

2019-05-13 15:07:51,900 : WARNING : scanning corpus to determine the number of features (consider setting `num_features` explicitly)
2019-05-13 15:07:51,902 : INFO : creating matrix with 24 documents and 10 features

小贴士:

similarities.MatrixSimilarity 这个类会将所有参与检索的语料一股脑的载入到本地内存中。比如，当使用该类时，某个包含百万级文档的语料库将以256维的LSI向量空间载入到只有2GB内存的电脑中，这很可能会引起电脑的卡顿，甚至直接黑屏。。。

如果没有2GB的可用RAM，但需要完成文档相似度检索这个任务，则需要使用 similarities.Similarity 这个类。此类通过将在本地上的多个文件（称为分片）之间拆 分索引，在固定内存中运行。它在内部使用 similarities.MatrixSimilarity 和 similarities.SparseMatrixSimilarity 这两个类，所以它仍然很快，尽管使用和理解稍微复杂一些。

索引持久化（Index Persistency）可以通过两个标准的函数 --- save（）和load（）来完成：

```
index.save(os.path.join(TEMP_FOLDER, '查询.index'))
index = similarities.MatrixSimilarity.load(os.path.join(TEMP_FOLDER, '查询.inde
```

2019-05-13 15:07:57,846 : INFO : saving MatrixSimilarity object under C:\Users\hp\AppData\Local\Temp\查询.index, separately None
2019-05-13 15:07:57,848 : INFO : saved C:\Users\hp\AppData\Local\Temp\查询.index
2019-05-13 15:07:57,850 : INFO : loading MatrixSimilarity object from C:\Users\hp\AppData\Local\Temp\查询.index
2019-05-13 15:07:57,852 : INFO : loaded C:\Users\hp\AppData\Local\Temp\查询.index

上述操作对于所有相似性索引类（similarities.Similarity，similarities.MatrixSimilarity 和 similarities.SparseMatrixSimilarity）都是一样的。如果你不确定使用哪个类来建立查询模型，那就请使用 similarities.Similarity，因为它是最具扩展性的版本，并且它还支持后续向索引中添加更多的文档。

三、实施查询（Performing Queries）

对给定查询文档 --- “**文本挖掘在舆情口碑挖掘中的作用很大**” 进行相似度查询，结果按匹配语句的相似度降序进行排序：

```
sims = index[vec_lsi]
result = [(docs[i[0]],i[1]) for i in enumerate(sims)]            # 对检索语料库进
pprint(sorted(result ,key=lambda x: x[1],reverse=True))          # 每个查询结果的
```

[('以《大秦帝国之崛起》为例，来谈大数据舆情分析和文本挖掘', 0.82118684),
('数据挖掘实操 | 用文本挖掘剖析近5万首《全唐诗》', 0.7825377),
('文本挖掘从小白到精通（二）---语料库和词向量空间', 0.64254516),
('文本挖掘从小白到精通（一）---语料、向量空间和模型的概念', 0.581834),
('从社交媒体传播和文本挖掘角度解读《欢乐颂2》', 0.5602435),
('用大数据文本挖掘，来洞察“共享单车”的行业现状及走势', 0.5551236),
('文本挖掘从小白到精通（三）---主题模型和文本数据转换', 0.5421253),
('以虎嗅网4W+文章的文本挖掘为例，展现数据分析的一整套流程', 0.52718097),
('傅园慧和她的“洪荒之力”的大数据舆情分析', 0.5001124),
('不懂数理和编程，如何运用免费的大数据工具获得行业洞察？', 0.2191386),
('干货 | 作为一个合格的“增长黑客”，你还得重视外部数据的分析！', 0.17165673),
('数据运营|数据分析中，文本分析远比数值型分析重要！（上）', 0.1559416),
('Social Listening和传统市场调研的关系是怎样的？', 0.0628911),
('如何利用Social Listening从社会化媒体中“提炼”有价值的信息？', 0.0553093),
('揭开微博转发传播的规律：以“人民日报”发布的G20文艺晚会微博为例', 0.04531411),
('【新媒体运营实操】如何做出一个精美的个性化词云以哈尔滨冰雪大世界旅游的传播效应为例，谈数据新闻可视化的“魅惑”', 0.040802844),
('当数据分析遭遇心理动力学：用户深层次的情感需求浮出水面', -0.033479266),
('万字干货 | 10款数据分析“工具”，助你成为新媒体运营领域的“增长黑客”', -0.054246806),
('文本分类算法集锦，从小白到大牛，附代码注释和训练语料', -0.0704931),
('数据运营实操 | 如何运用数据分析对某个试运营项目进行“无死角”的复盘？', -0.07056374),
('如何用数据分析，搞定新媒体运营的定位和内容初始化？', -0.08224367),
('如何利用微信后台数据优化微信运营', -0.15745023),
('如何用聚类分析进行企业公众号的内容优化', -0.2176962),
('写给迷茫的你：如何运用运营思维规划自己的职业发展路径？', -0.25396234)]

余弦度量（Cosine Measure）返回的值域为<-1,1>，数值越大，表示和检索文档的相似度越高，比如返回结果中排行第一的语句的相似度最高，为**0.82118684**.

看到这里，有些细心的同学可能会问，以下语句并没有包含查询文档中的关键词，应该不会被标准的布尔全文搜索查找到，但为什么还是会被检索出来呢？比如：

- 以哈尔滨冰雪大世界旅游的传播效应为例，谈数据新闻可视化的“魅惑”
- 微博转发传播的规律：以“人民日报”发布的G20文艺晚会微博为例
- 利用Social Listening从社会化媒体中“提炼”有价值的信息？
- 干货 | 作为一个合格的“增长黑客”，你还得重视外部数据的分析！

- 如何用聚类分析进行企业公众号的内容优化
- 数据运营|数据分析中，文本分析远比数值型分析重要！（上）

这是因为，在应用LSI对语料进行建模后，这些文档间的潜在语义关联被发掘出来了，只要与主题相关的语句，即使其中包含的词汇不完全一样，比如"Social Listening"、"舆情"、"传播效应"这3个词汇，虽然直观上看起来没啥相关性，但因为内涵相近，都是涉及社交媒体数据挖掘，所以包含它们的语句也会在相似结果中进行呈现。事实上，这种语义的推断和概括是我们应用之前提及的各类文本数据转换并进行主题建模的根本原因。

针对基于LSI进行文档相似度检索，还可以优化的方面:

> **（1 ）对语义的挖掘还不够。这个相似度查询还比较简陋，是硬匹配（非命中匹配关键词才行），意义相近但说法不一样的词汇的相似性还不能很好的度量，这在后面基于word2vec或者doc2vec的模型可以很好的解决;**
> **（2） 速度有待优化。当检索的文档相当大时，这个检索系统的效率就会非常低，这时可以采用Annoy搜索算法(Approximate Nearest Neighbors Oh Yeah)来缓解。**

关于上面提到的两个待优化的方面，且听笔者在后续的文章中细说~

推荐阅读

这个NLP工具，玩得根本停不下来

从数据到模型，你可能需要1篇详实的pytorch踩坑指南

如何让Bert在finetune小数据集时更"稳"一点

模型压缩实践系列之——bert-of-theseus，一个非常亲民的bert压缩方法

征稿启示| 200元稿费+5000DBC（价值20个小时GPU算力）

文本自动摘要任务的"不完全"心得总结番外篇——submodular函数优化

Node2Vec 论文+代码笔记

模型压缩实践收尾篇——模型蒸馏以及其他一些技巧实践小结

中文命名实体识别工具（NER）哪家强?

学自然语言处理，其实更应该学好英语

斯坦福大学NLP组Python深度学习自然语言处理工具Stanza试用

关于AINLP

AINLP 是一个有趣有AI的自然语言处理社区，专注于 AI、NLP、机器学习、深度学习、推荐算法等相关技术的分享，主题包括文本摘要、智能问答、聊天机器人、机器翻译、自动生成、知识图谱、预训练模型、推荐系统、计算广告、招聘信息、求职经验分享等，欢迎关注！加技术交流群请添加AINLPer(id：ainlper)，备注工作/研究方向+加群目的。

阅读至此了，分享、点赞、在看三选一吧🙏