知乎 首发于 银河系资讯

使用Gensim进行主题建模(一)

银河1号
一起发现全球深度好文

关注他

主题建模是一种从大量文本中提取隐藏主题的技术。Latent Dirichlet Allocation（LDA）是一种流行的主题建模算法，在Python的Gensim包中具有出色的实现。然而，挑战在于如何提取清晰，隔离和有意义的高质量主题。这在很大程度上取决于文本预处理的质量以及找到最佳主题数量的策略。本教程试图解决这两个问题。

内容

1.简介

件等。

了解人们在谈论什么并理解他们的问题和意见对于企业，管理者和政治活动来说非常有价值。并且很难人工阅读如此大数据量的文本并识别主题。

因此，需要一种自动算法，该算法可以读取文本文档并自动输出所讨论的主题。

在本教程中，我们将采用'20新闻组'数据集的真实示例，并使用LDA提取自然讨论的主题。

我将使用Gensim包中的Latent Dirichlet Allocation（LDA）以及Mallet的实现（通过Gensim）。Mallet有效地实现了LDA。众所周知，它可以更快地运行并提供更好的主题隔离。

我们还将提取每个主题的数量和百分比贡献，以了解主题的重要性。

让我们开始！

2.先决条件 - 下载nltk停用词和spacy模型

我们需要来自NLTK的 `stopwords` 和spacy的 en 模型进行词形还原。

```
# Run in python console
import nltk; nltk.download('stopwords')

# Run in terminal or command prompt
python3 -m spacy download en
```

3.导入包

在本教程中使用的核心包 `re` ， `gensim` ， `spacy` 和 `pyLDAvis` 。除此之外，我们还将使用 `matplotlib` ， `numpy` 以及 `pandas` 数据处理和可视化。让我们导入它们。

```
import re
import numpy as np
import pandas as pd
from pprint import pprint

# Gensim
import gensim
import gensim.corpora as corpora
from gensim.utils import simple_preprocess
from gensim.models import CoherenceModel

# spacy for lemmatization
import spacy

# Plotting tools
import pyLDAvis
import pyLDAvis.gensim  # don't skip this
import matplotlib.pyplot as plt
%matplotlib inline

# Enable logging for gensim - optional
import logging
logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s', level=

import warnings
warnings.filterwarnings("ignore",category=DeprecationWarning)
```

LDA的主题建模方法是将每个文档视为一定比例的主题集合。并且每个主题作为关键字的集合，再次以一定比例。

一旦您为算法提供了主题数量，它就会重新排列文档中的主题分布和主题内的关键字分布，以获得主题 - 关键字分布的良好组合。

当我说主题时，它实际上是什么以及如何表示?

一个主题只不过是典型代表的主导关键词集合。只需查看关键字，您就可以确定主题的内容。

以下是获得良好隔离主题的关键因素：

1. 文本处理的质量。
2. 文本谈论的各种主题。
3. 主题建模算法的选择。
4. 提供给算法的主题数量。
5. 算法调整参数。

5.准备关键词

我们已经下载了停用词。让我们导入它们并使其可用 `stop_words` 。

```
# NLTK Stop words
from nltk.corpus import stopwords
stop_words = stopwords.words('english')
stop_words.extend(['from', 'subject', 're', 'edu', 'use'])
```

6.导入新闻组数据

我们将使用20-Newsgroups数据集进行此练习。此版本的数据集包含来自20个不同主题的大约11k个新闻组帖子。这可以作为newsgroups.json使用。

这是使用导入的 `pandas.read_json` ，结果数据集有3列，如图所示。

```
# Import Dataset
df = pd.read_json('https://raw.githubusercontent.com/selva86/datasets/master/ne
print(df.target_names.unique())
df.head()
```

```
 'rec.sport.baseball' 'rec.sport.hockey' 'sci.electronics' 'sci.space'
 'talk.politics.misc' 'sci.med' 'talk.politics.mideast'
 'soc.religion.christian' 'comp.windows.x' 'comp.sys.ibm.pc.hardware'
 'talk.politics.guns' 'talk.religion.misc' 'sci.crypt']
```

content	target	target_names
From: lerxst@wam.umd.edu (where's my thing)\nSubject: WHAT car is this!?\nNntp-Posting-Host: rac...	7	rec.autos
From: guykuo@carson.u.washington.edu (Guy Kuo)\nSubject: SI Clock Poll - Final Call\nSummary: Fi...	4	comp.sys.mac.hardware
From: irwin@cmptrc.lonestar.org (Irwin Arnstein)\nSubject: Re: Recommendation on Duc\nSummary: W...	8	rec.motorcycles
From: tchen@magnus.acs.ohio-state.edu (Tsung-Kun Chen)\nSubject: ** Software forsale (lots) **\n...	6	misc.forsale
From: dabl2@nlm.nih.gov (Don A.B. Lindbergh)\nSubject: Diamond SS24X, Win 3.1, Mouse cursor\nOrg...	2	comp.os.ms-windows.misc
From: a207706@moe.dseg.ti.com (Robert Loper)\nSubject: Re: SHO and SC\nNntp-Posting-Host: sun278...	7	rec.autos
From: kimman@magnus.acs.ohio-state.edu (Kim Richard Man)\nSubject: SyQuest 44M cartrifge FORSALE...	6	misc.forsale
From: kwilson@casbah.acns.nwu.edu (Kirtley Wilson)\nSubject: Mirosoft Office Package\nArticle-I....	2	comp.os.ms-windows.misc
Subject: Re: Don't more innocents die without the death penalty?\nFrom: bobbe@vice.ICO.TEK.COM (...	0	alt.atheism
From: livesey@solntze.wpd.sgi.com (Jon Livesey)\nSubject: Re: Genocide is Caused by Atheism\nOrg...	0	alt.atheism
From: dls@aeg.dsto.gov.au (David Silver)\nSubject: Re: Fractal Generation of Clouds\nOrganizatio...	1	comp.graphics
Subject: Re: Mike Francesa's 1993 Predictions\nFrom: gajarsky@pilot.njin.net (Bob Gajarsky - Hob...	9	rec.sport.baseball
From: jet@netcom.Netcom.COM (J. Eric Townsend)\nSubject: Re: Insurance and lotsa points...\nIn-R...	8	rec.motorcycles
From: gld@cunixb.cc.columbia.edu (Gary L Dare)\nSubject: Re: ABC coverage\nNntp-Posting-Host: cu...	10	rec.sport.hockey
From: sehari@iastate.edu (Babak Sehari)\nSubject: Re: How to the disks copy protected.\nOriginat...	12	sci.electronics

20个新闻组数据集

7.删除电子邮件和换行符

正如您所看到的那样，有许多电子邮件，换行符和额外空间非常分散注意力。让我们使用正则表达式摆脱它们。

```
# Convert to list
data = df.content.values.tolist()

# Remove Emails
data = [re.sub('\S*@\S*\s?', '', sent) for sent in data]

# Remove new line characters
data = [re.sub('\s+', ' ', sent) for sent in data]

# Remove distracting single quotes
data = [re.sub("\'", "", sent) for sen
```

```
['From: (wheres my thing) Subject: WHAT car is this!? Nntp-Posting-Host: '
 'rac3.wam.umd.edu Organization: University of Maryland, College Park Lines: '
 '15 I was wondering if anyone out there could enlighten me on this car I saw '
 'the other day. It was a 2-door sports car, looked to be from the late 60s/ '
 'early 70s. It was called a Bricklin. The doors were really small. In '
 'addition, the front bumper was separate from the rest of the body. This is '
 'all I know.  (..truncated..)]
```

删除电子邮件和额外空格后，文本仍然看起来很乱。它尚未准备好让LDA消费。您需要通过标记化将每个句子分解为单词列表，同时清除过程中的所有杂乱文本。

Gensim对此很有帮助 `simple_preprocess` 。

8.标记单词和清理文本

让我们将每个句子标记为一个单词列表，完全删除标点符号和不必要的字符。

Gensim对此很有帮助 `simple_preprocess()` 。此外，我已经设置 `deacc=True` 删除标点符号。

```
def sent_to_words(sentences):
    for sentence in sentences:
        yield(gensim.utils.simple_preprocess(str(sentence), deacc=True))  # dea

data_words = list(sent_to_words(data))

print(data_words[:1])
[['from', 'wheres', 'my', 'thing', 'subject', 'what', 'car', 'is', 'this', 'nnt
```

9.创建Bigram和Trigram模型

Bigrams是文档中经常出现的两个词。Trigrams经常出现3个单词。

我们示例中的一些示例是：'front_bumper'，'oil_leak'，'maryland_college_park'等。

Gensim的 `Phrases` 模型可以构建和实现bigrams，trigrams，quadgrams等。两个重要的论点 `Phrases` 是 `min_count` 和 `threshold` 。这些参数的值越高，将单词组合成双字母组的难度就越大。

```
trigram = gensim.models.Phrases(bigram[data_words], threshold=100)

# Faster way to get a sentence clubbed as a trigram/bigram
bigram_mod = gensim.models.phrases.Phraser(bigram)
trigram_mod = gensim.models.phrases.Phraser(trigram)

# See trigram example
print(trigram_mod[bigram_mod[data_words[0]]])
['from', 'wheres', 'my', 'thing', 'subject', 'what', 'car', 'is', 'this', 'nntp
```

10.删除停用词，制作Bigrams和Lemmatize

双胞胎模型准备就绪。让我们定义函数来删除停用词，制作双字母组合和词形还原并按顺序调用它们。

```
# Define functions for stopwords, bigrams, trigrams and lemmatization
def remove_stopwords(texts):
    return [[word for word in simple_preprocess(str(doc)) if word not in stop_w

def make_bigrams(texts):
    return [bigram_mod[doc] for doc in texts]

def make_trigrams(texts):
    return [trigram_mod[bigram_mod[doc]] for doc in texts]

def lemmatization(texts, allowed_postags=['NOUN', 'ADJ', 'VERB', 'ADV']):
    """https://spacy.io/api/annotation"""
    texts_out = []
    for sent in texts:
        doc = nlp(" ".join(sent))
        texts_out.append([token.lemma_ for token in doc if token.pos_ in allowe
    return texts_out
```

我们按顺序调用这些函数。

```
# Remove Stop Words
data_words_nostops = remove_stopwords(data_words)

# Form Bigrams
```

```
# Initialize spacy 'en' model, keeping only tagger component (for efficiency)
# python3 -m spacy download en
nlp = spacy.load('en', disable=['parser', 'ner'])

# Do lemmatization keeping only noun, adj, vb, adv
data_lemmatized = lemmatization(data_words_bigrams, allowed_postags=['NOUN', 'A

print(data_lemmatized[:1])
[['where', 's', 'thing', 'car', 'nntp_post', 'host', 'rac_wam', 'umd', 'organi
```

11.创建主题建模所需的词典和语料库

LDA主题模型的两个主要输入是字典（ id2word ）和语料库。让我们创造它们。

```
# Create Dictionary
id2word = corpora.Dictionary(data_lemmatized)

# Create Corpus
texts = data_lemmatized

# Term Document Frequency
corpus = [id2word.doc2bow(text) for text in texts]

# View
print(corpus[:1])
[[(0, 1), (1, 2), (2, 1), (3, 1), (4, 1), (5, 1), (6, 5), (7, 1), (8, 1), (9, 1
```

Gensim为文档中的每个单词创建一个唯一的ID。上面显示的产生的语料库是（word_id, word_frequency）的映射。

例如，上面的（0,1）暗示，单词id 0在第一个文档中出现一次。同样，单词id 1出现两次，依此类推。

这用作LDA模型的输入。

如果要查看给定id对应的单词，请将id作为键传递给字典。

```
id2word[0]
'addition'
```

```
# Human readable format of corpus (term-frequency)
[[(id2word[id], freq) for id, freq in cp] for cp in corpus[:1]]
[[('addition', 1),
  ('anyone', 2),
  ('body', 1),
  ('bricklin', 1),
  ('bring', 1),
  ('call', 1),
  ('car', 5),
  ('could', 1),
  ('day', 1),
  ('door', 2),
  ('early', 1),
  ('engine', 1),
  ('enlighten', 1),
  ('front_bumper', 1),
  ('maryland_college', 1),
  (..truncated..)]]
```

好吧，如果不进一步离题，让我们重新回到正轨，进行下一步：构建主题模型。

12.构建主题模型

我们拥有培训LDA模型所需的一切。除语料库和字典外，您还需要提供主题数量。

除此之外， `alpha` 还有 `eta` 影响主题稀疏性的超参数。根据Gensim文档，默认为1.0 / num_topics之前。

`chunksize` 是每个训练块中使用的文档数。 `update_every` 确定应更新模型参数的频率，以及 `passes` 培训通过的总数。

```
# Build LDA model
lda_model = gensim.models.ldamodel.LdaModel(corpus=corpus,
                                            id2word=id2word,
                                            num_topics=20,
                                            random_state=100,
                                            update_every=1,
                                            chunksize=100,
```

13.查看LDA模型中的主题

上述LDA模型由20个不同的主题构建，其中每个主题是关键字的组合，并且每个关键字对主题贡献一定的权重。

您可以看到每个主题的关键字以及每个关键字的权重（重要性）， `lda_model.print_topics()` 如下所示。

```
# Print the Keyword in the 10 topics
pprint(lda_model.print_topics())
doc_lda = lda_model[corpus]
[(0,
  '0.016*"car" + 0.014*"power" + 0.010*"light" + 0.009*"drive" + 0.007*"mount"
  '+ 0.007*"controller" + 0.007*"cool" + 0.007*"engine" + 0.007*"back" + '
  '0.006*"turn"'),
 (1,
  '0.072*"line" + 0.066*"organization" + 0.037*"write" + 0.032*"article" + '
  '0.028*"university" + 0.027*"nntp_post" + 0.026*"host" + 0.016*"reply" + '
  '0.014*"get" + 0.013*"thank"'),
 (2,
  '0.017*"patient" + 0.011*"study" + 0.010*"slave" + 0.009*"wing" + '
  '0.009*"disease" + 0.008*"food" + 0.008*"eat" + 0.008*"pain" + '
  '0.007*"treatment" + 0.007*"syndrome"'),
 (3,
  '0.013*"key" + 0.009*"use" + 0.009*"may" + 0.007*"public" + 0.007*"system" +
  '0.007*"order" + 0.007*"government" + 0.006*"state" + 0.006*"provide" + '
  '0.006*"law"'),
 (4,
  '0.568*"ax" + 0.007*"rlk" + 0.005*"tufts_university" + 0.004*"ei" + '
  '0.004*"m" + 0.004*"vesa" + 0.004*"differential" + 0.004*"chz" + 0.004*"lk"
  '+ 0.003*"weekly"'),
 (5,
  '0.029*"player" + 0.015*"master" + 0.015*"steven" + 0.009*"tor" + '
  '0.009*"van" + 0.008*"king" + 0.008*"scripture" + 0.007*"cal" + '
  '0.007*"helmet" + 0.007*"det"'),
 (6,
  '0.028*"system" + 0.020*"problem" + 0.019*"run" + 0.018*"use" + 0.016*"work"
  '+ 0.015*"do" + 0.013*"window" + 0.013*"driver" + 0.013*"bit" + 0.012*"set"'
 (7,
  '0.017*"israel" + 0.011*"israeli" +
```

```
(8,
 '0.018*"money" + 0.018*"year" + 0.016*"pay" + 0.012*"car" + 0.010*"drug" + '
 '0.010*"president" + 0.009*"rate" + 0.008*"face" + 0.007*"license" + '
 '0.007*"american"'),
(9,
 '0.028*"god" + 0.020*"evidence" + 0.018*"christian" + 0.012*"believe" + '
 '0.012*"reason" + 0.011*"faith" + 0.009*"exist" + 0.008*"bible" + '
 '0.008*"religion" + 0.007*"claim"'),
(10,
 '0.030*"physical" + 0.028*"science" + 0.012*"direct" + 0.012*"st" + '
 '0.012*"scientific" + 0.009*"waste" + 0.009*"jeff" + 0.008*"cub" + '
 '0.008*"brown" + 0.008*"msg"'),
(11,
 '0.016*"wire" + 0.011*"keyboard" + 0.011*"md" + 0.009*"pm" + 0.008*"air" + '
 '0.008*"input" + 0.008*"fbi" + 0.007*"listen" + 0.007*"tube" + '
 '0.007*"koresh"'),
(12,
 '0.016*"motif" + 0.014*"serial_number" + 0.013*"son" + 0.013*"father" + '
 '0.011*"choose" + 0.009*"server" + 0.009*"event" + 0.009*"value" + '
 '0.007*"collin" + 0.007*"prediction"'),
(13,
 '0.098*"_" + 0.043*"max" + 0.015*"dn" + 0.011*"cx" + 0.009*"eeg" + '
 '0.008*"gateway" + 0.008*"c" + 0.005*"mu" + 0.005*"mr" + 0.005*"eg"'),
(14,
 '0.024*"book" + 0.009*"april" + 0.007*"group" + 0.007*"page" + '
 '0.007*"new_york" + 0.007*"iran" + 0.006*"united_state" + 0.006*"author" + '
 '0.006*"include" + 0.006*"club"'),
(15,
 '0.020*"would" + 0.017*"say" + 0.016*"people" + 0.016*"think" + 0.014*"make"
 '+ 0.014*"go" + 0.013*"know" + 0.012*"see" + 0.011*"time" + 0.011*"get"'),
(16,
 '0.026*"file" + 0.017*"program" + 0.012*"window" + 0.012*"version" + '
 '0.011*"entry" + 0.011*"software" + 0.011*"image" + 0.011*"color" + '
 '0.010*"source" + 0.010*"available"'),
(17,
 '0.027*"game" + 0.027*"team" + 0.020*"year" + 0.017*"play" + 0.016*"win" + '
 '0.010*"good" + 0.009*"season" + 0.008*"fan" + 0.007*"run" + 0.007*"score"'),
(18,
 '0.036*"drive" + 0.024*"card" + 0.020*"mac" + 0.017*"sale" + 0.014*"cpu" + '
 '0.010*"price" + 0.010*"disk" + 0.010*"board" + 0.010*"pin" + 0.010*"chip"'),
(19,
 '0.030*"space" + 0.010*"sphere" + 0.
```

怎么解释这个?

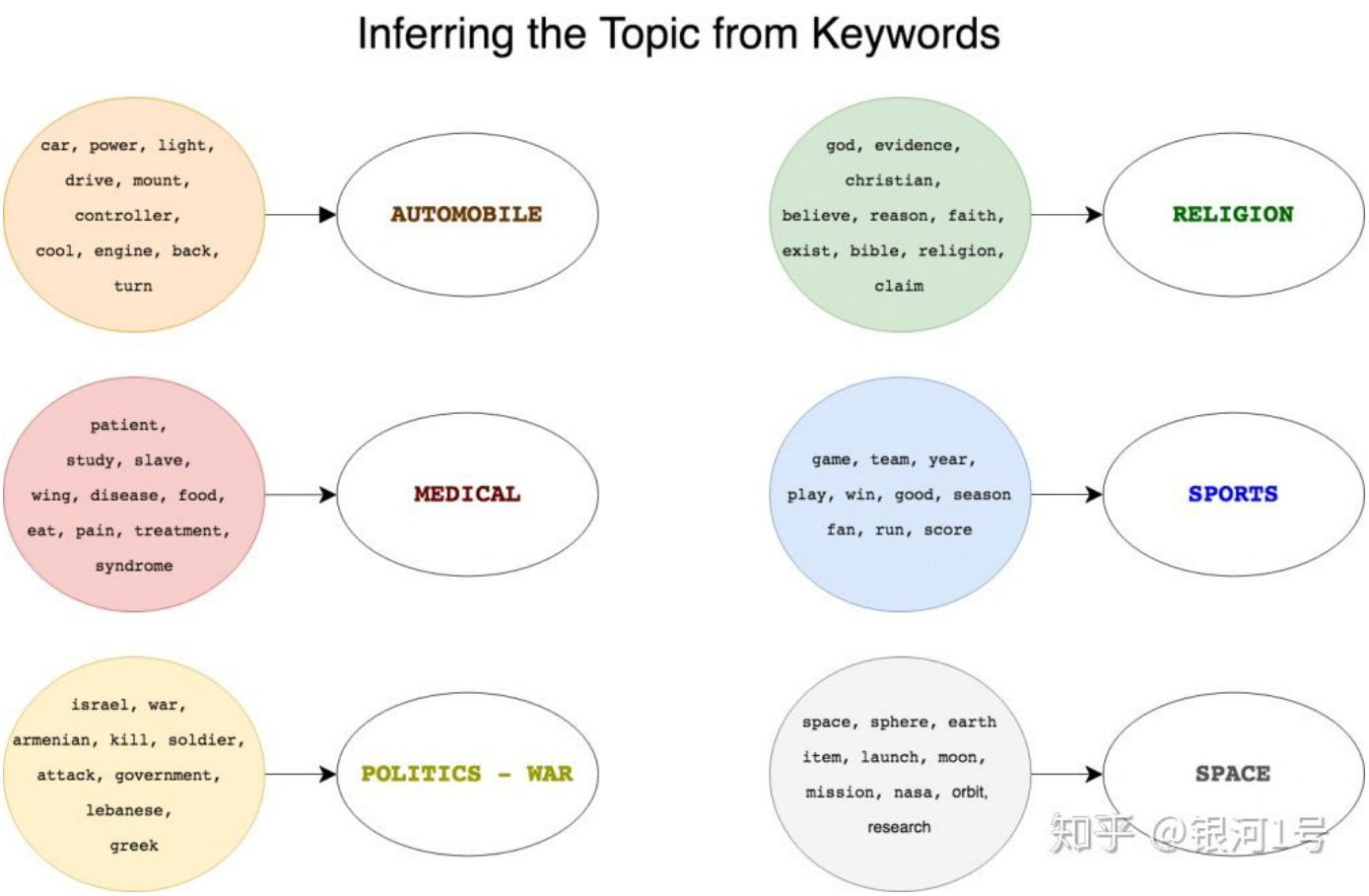
主题0表示为_0.016 *“car”+ 0.014* “power”+ 0.010 *“light”+ 0.009* “drive”+ 0.007 *“mount”+ 0.007*“controller”+ 0.007 *“cool”+ 0.007* “engine”+ 0.007 *“回“+'0.006* "转“。

这意味着贡献这个主题的前10个关键词是：'car'，'power'，'light'等等，主题0上'car'的重量是0.016。

权重反映了关键字对该主题的重要程度。

看看这些关键词，您能猜出这个主题是什么吗？您可以将其概括为“汽车”或“汽车”。

同样，您是否可以浏览剩余的主题关键字并判断主题是什么?



从关键字推断主题

14.计算模型困惑和一致性分数

```
# Compute Perplexity
print('\nPerplexity: ', lda_model.log_perplexity(corpus))  # a measure of how g

# Compute Coherence Score
coherence_model_lda = CoherenceModel(model=lda_model, texts=data_lemmatized, di
coherence_lda = coherence_model_lda.get_coherence()
print('\nCoherence Score: ', coherence_lda)
Perplexity:  -8.86067503009

Coherence Score:  0.532947587081
```

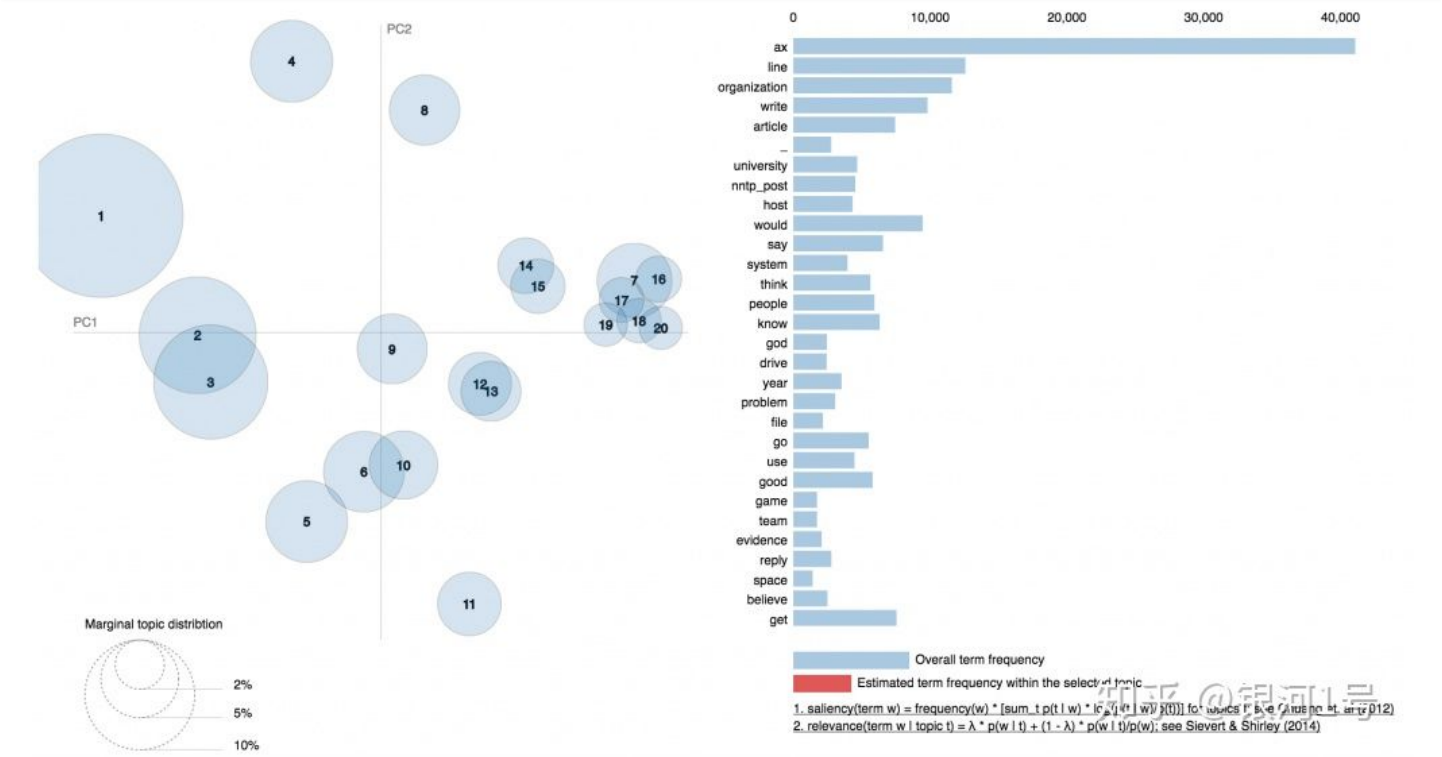
你有一个0.53的连贯性得分。

15.可视化主题 - 关键字

现在已经构建了LDA模型，下一步是检查生成的主题和关联的关键字。没有比pyLDAvis包的交互式图表更好的工具，并且设计为与jupyter笔记本一起使用。

```
# Visualize the topics
pyLDAvis.enable_notebook()
vis = pyLDAvis.gensim.prepare(lda_model, corpus, id2word)
vis
```

知乎 首发于 银河系资讯



PYLDAVIS输出

那么如何推断pyLDAvis的输出呢?

左侧图中的每个气泡代表一个主题。泡沫越大，该主题就越普遍。

一个好的主题模型将在整个图表中分散相当大的非重叠气泡，而不是聚集在一个象限中。

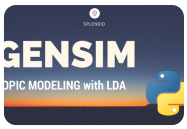
具有太多主题的模型通常会有许多重叠，小尺寸的气泡聚集在图表的一个区域中。

好吧，如果将光标移动到其中一个气泡上，右侧的单词和条形将会更新。这些单词是构成所选主题的显着关键字。

我们已经成功构建了一个好看的主题模型。

鉴于我们之前对文档中自然主题数量的了解，找到最佳模型非常简单。

Topic Modeling with Gensim (Python)
www.machinelearningplus.com



知乎 首发于 银河系资讯

lda
公众号:银河系1号
联系邮箱：public@space-explore.com
(未经同意，请勿转载)

发布于 2019-04-12

「真诚赞赏，手留余香」

赞赏

还没有人赞赏，快来当第一个赞赏的人吧！

LDA 自然语言处理 Python

文章被以下专栏收录

银河系资讯
发布前沿的科技文章

关注专栏

推荐阅读

NLP实战：用主题
评论（附Python代

使用Gensim进行主题建模（二）

在上一篇文章中，我们将使用Mallet版本的LDA算法对此模型进行改进，然后我们将重点介绍如何在

知乎 首发于 **银河系资讯**

1 条评论 切换为时间排序

写下你的评论...

韬韬锅 2019-11-28

如果写出引用出处，可能更好一点machinelearningplus.com...

赞