各种花里胡哨的NL编码

2020年6月7日

15:23

# 资料 ([页首](onenote:#各种花里胡哨的NL编码&section-id={96A5A6B9-06D6-4B2B-884F-F2144848BB51}&page-id={7D7D0CC0-63BF-4DFB-9373-C732732D1FA0}&object-id={F1E24A57-5CE6-40A2-8442-D349A987FB90}&F&base-path=https://d.docs.live.net/04abe269f00f36a1/文档/数据挖掘/NLP比赛.one))

## [sklearn.feature\_extraction.text](https://scikit-learn.org/0.18/modules/classes.html#module-sklearn.feature_extraction.text) ([页首](onenote:#各种花里胡哨的NL编码&section-id={96A5A6B9-06D6-4B2B-884F-F2144848BB51}&page-id={7D7D0CC0-63BF-4DFB-9373-C732732D1FA0}&object-id={F1E24A57-5CE6-40A2-8442-D349A987FB90}&F&base-path=https://d.docs.live.net/04abe269f00f36a1/文档/数据挖掘/NLP比赛.one))

# [PROJ.4学习——初识PROJ（普通坐标系转为地理的经纬度）](https://blog.csdn.net/weixin_34148340/article/details/94033366) ([页首](onenote:#各种花里胡哨的NL编码&section-id={96A5A6B9-06D6-4B2B-884F-F2144848BB51}&page-id={7D7D0CC0-63BF-4DFB-9373-C732732D1FA0}&object-id={F1E24A57-5CE6-40A2-8442-D349A987FB90}&F&base-path=https://d.docs.live.net/04abe269f00f36a1/文档/数据挖掘/NLP比赛.one))

## 相关资料 ([页首](onenote:#各种花里胡哨的NL编码&section-id={96A5A6B9-06D6-4B2B-884F-F2144848BB51}&page-id={7D7D0CC0-63BF-4DFB-9373-C732732D1FA0}&object-id={F1E24A57-5CE6-40A2-8442-D349A987FB90}&F&base-path=https://d.docs.live.net/04abe269f00f36a1/文档/数据挖掘/NLP比赛.one))

[地理位置对应的代码速查](https://mygeodata.cloud/cs2cs/)

## 秒懂的实现步骤 ([页首](onenote:#各种花里胡哨的NL编码&section-id={96A5A6B9-06D6-4B2B-884F-F2144848BB51}&page-id={7D7D0CC0-63BF-4DFB-9373-C732732D1FA0}&object-id={F1E24A57-5CE6-40A2-8442-D349A987FB90}&F&base-path=https://d.docs.live.net/04abe269f00f36a1/文档/数据挖掘/NLP比赛.one))

* + 一、在[地理位置对应的代码速查](https://mygeodata.cloud/cs2cs/)中找到proj\_text或者epsg，通过以下格式实现proj\_code

"cs2cs +init=epsg:4326 +to +init=epsg:25832"

"cs2cs proj\_text +to proj\_text"

举例 Xian1980

"+proj=tmerc +lat\_0=0 +lon\_0=120 +k=1 +x\_0=500000 +y\_0=0 +a=6378140 +b=6356755.288157528 +units=m +no\_defs "

"EPSG:2385"

* + 二、实现代码

from pyproj import Proj

def transform\_xy2xy(df):

x = df['x'].values

y = df['y'].values

p = Proj(""EPSG:2385")

df['x'], df['y'] = p(x, y, inverse=True)#inverse=True表示把以米为单位的直角坐标系转化为经纬度

return df

# [Geohash算法原理及实现](https://www.jianshu.com/p/2fd0cf12e5ba)（经纬度编码） ([页首](onenote:#各种花里胡哨的NL编码&section-id={96A5A6B9-06D6-4B2B-884F-F2144848BB51}&page-id={7D7D0CC0-63BF-4DFB-9373-C732732D1FA0}&object-id={F1E24A57-5CE6-40A2-8442-D349A987FB90}&F&base-path=https://d.docs.live.net/04abe269f00f36a1/文档/数据挖掘/NLP比赛.one))

## 相关资料： ([页首](onenote:#各种花里胡哨的NL编码&section-id={96A5A6B9-06D6-4B2B-884F-F2144848BB51}&page-id={7D7D0CC0-63BF-4DFB-9373-C732732D1FA0}&object-id={F1E24A57-5CE6-40A2-8442-D349A987FB90}&F&base-path=https://d.docs.live.net/04abe269f00f36a1/文档/数据挖掘/NLP比赛.one))

[Github源码](https://github.com/vinsci/geohash)

[Geohash的python实现](https://www.jianshu.com/p/f2939ab349da)

## 代码实现 ([页首](onenote:#各种花里胡哨的NL编码&section-id={96A5A6B9-06D6-4B2B-884F-F2144848BB51}&page-id={7D7D0CC0-63BF-4DFB-9373-C732732D1FA0}&object-id={F1E24A57-5CE6-40A2-8442-D349A987FB90}&F&base-path=https://d.docs.live.net/04abe269f00f36a1/文档/数据挖掘/NLP比赛.one))

from geohash import \*

encode(lat,lon,precision=12)

decode(geohash)

neighbors(geohash)#返回8个相邻位置的哈希编码

bbox(geohash)#返回左上和右下的位置

## 认识 ([页首](onenote:#各种花里胡哨的NL编码&section-id={96A5A6B9-06D6-4B2B-884F-F2144848BB51}&page-id={7D7D0CC0-63BF-4DFB-9373-C732732D1FA0}&object-id={F1E24A57-5CE6-40A2-8442-D349A987FB90}&F&base-path=https://d.docs.live.net/04abe269f00f36a1/文档/数据挖掘/NLP比赛.one))

* + GeoHash表示的并不是一个点，而是一个矩形区域
  + GeoHash编码的前缀可以表示更大的区域。例如wx4g0ec1，它的前缀wx4g0e表示包含编码wx4g0ec1在内的更大范围。 这个特性可以用于附近地点搜索
  + 编码越长，表示的范围越小，位置也越精确

geohash length 
1 
2 
3 
4 
5 
6 
7 
8 
lat bits 
2 
5 
7 
10 
12 
15 
17 
20 
Ing bits 
3 
5 
8 
10 
13 
15 
18 
20 
lat error 
±2.8 
±0.70 
±0.087 
±0.022 
±0.0027 
±0.00068 
±0.000085 
Ing error 
± 70 
±0.18 
±0.022 
±0.0055 
±0.00068 
± 0.00017 
km error 
±2500 
±630 
± 20 
±2.4 
±0.61 
±0.076 
±0.019 

# [三种文本特征提取（TF-IDF/Word2Vec/CountVectorizer）](https://blog.csdn.net/ximibbb/article/details/79264574) ([页首](onenote:#各种花里胡哨的NL编码&section-id={96A5A6B9-06D6-4B2B-884F-F2144848BB51}&page-id={7D7D0CC0-63BF-4DFB-9373-C732732D1FA0}&object-id={F1E24A57-5CE6-40A2-8442-D349A987FB90}&F&base-path=https://d.docs.live.net/04abe269f00f36a1/文档/数据挖掘/NLP比赛.one))

# [简单介绍四种文本特征表示](https://tianchi.aliyun.com/notebook-ai/detail?spm=5176.12586969.1002.45.6406111asyOjjr&postId=118254)

* + One-hot:每个字确定一个编号
  + ag of Words（词袋表示),也称为Count Vectors:直接统计每个字出现的次数
  + N-gram:

与Count Vectors类似，不过加入了相邻单词组合成为新的单词，并进行计数。

如果N取值为2，则句子1和句子2就变为：

句子1：我爱 爱北 北京 京天 天安 安门

句子2：我喜 喜欢 欢上 上海

* + TF-IDF:相对Count Vectors，加入了词汇权重，滤除高频词汇的影响

都有一个缺点，得到的词向量维度很高。

## 相关资料 ([页首](onenote:#各种花里胡哨的NL编码&section-id={96A5A6B9-06D6-4B2B-884F-F2144848BB51}&page-id={7D7D0CC0-63BF-4DFB-9373-C732732D1FA0}&object-id={F1E24A57-5CE6-40A2-8442-D349A987FB90}&F&base-path=https://d.docs.live.net/04abe269f00f36a1/文档/数据挖掘/NLP比赛.one))

[Scikit-learn CountVectorizer与TfidfVectorizer](https://blog.csdn.net/The_lastest/article/details/79093407)

[scikit-learn：通过TruncatedSVD实现LSA（隐含语义分析）](https://blog.csdn.net/mmc2015/article/details/46867773)

[TruncatedSVD官方文档](https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.TruncatedSVD.html)

## [TfidfVectorizer](https://scikit-learn.org/0.18/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html#sklearn.feature_extraction.text.TfidfVectorizer)（TF-IDF词频－逆向文件频率） ([页首](onenote:#各种花里胡哨的NL编码&section-id={96A5A6B9-06D6-4B2B-884F-F2144848BB51}&page-id={7D7D0CC0-63BF-4DFB-9373-C732732D1FA0}&object-id={F1E24A57-5CE6-40A2-8442-D349A987FB90}&F&base-path=https://d.docs.live.net/04abe269f00f36a1/文档/数据挖掘/NLP比赛.one))

* + *资料 (*[*页首*](onenote:#各种花里胡哨的NL编码&section-id={96A5A6B9-06D6-4B2B-884F-F2144848BB51}&page-id={7D7D0CC0-63BF-4DFB-9373-C732732D1FA0}&object-id={F1E24A57-5CE6-40A2-8442-D349A987FB90}&F&base-path=https://d.docs.live.net/04abe269f00f36a1/文档/数据挖掘/NLP比赛.one)*)*

[sklearn: 利用TruncatedSVD做文本主题分析](https://blog.csdn.net/blmoistawinde/article/details/83446529?utm_medium=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-1.nonecase&depth_1-utm_source=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-1.nonecase)

* + [Python中的TfidfVectorizer参数解析](https://blog.csdn.net/laobai1015/article/details/80451371)
  + *基础(*[*页首*](onenote:#各种花里胡哨的NL编码&section-id={96A5A6B9-06D6-4B2B-884F-F2144848BB51}&page-id={7D7D0CC0-63BF-4DFB-9373-C732732D1FA0}&object-id={F1E24A57-5CE6-40A2-8442-D349A987FB90}&F&base-path=https://d.docs.live.net/04abe269f00f36a1/文档/数据挖掘/NLP比赛.one)*)*
    - 定义

TF(w)=(词w在文档中出现的次数)/(文档的总词数) #文档指的是这个文章，不是整个文章的集合！

IDF：逆向文件频率。#有些词在文本中频繁出现，但并不重要，也即信息量小

* + 使用

vectorizer = CountVectorizer() #构建一个计算词频（TF）的玩意儿，当然这里面不足是可以做这些

transformer = TfidfTransformer() #构建一个计算TF-IDF的玩意儿

tfidf = transformer.fit\_transform(vectorizer.fit\_transform(corpus)

#vectorizer.fit\_transform(corpus)将文本corpus输入，得到词频矩阵

#将这个矩阵作为输入，用transformer.fit\_transform(词频矩阵)得到TF-IDF权重矩阵

* + 对比Countvectorizer的优势

是一种在文本挖掘中广泛使用的特征向量化方法，它可以体现一个文档中词语在语料库中的重要程度。(对比Countvectorizer，训练文本的数量越多，TF-IDF就更有优势。）

* + *举例 与TruncatedSVD结合实现LSI/LSA（隐含语义分析） (*[*页首*](onenote:#各种花里胡哨的NL编码&section-id={96A5A6B9-06D6-4B2B-884F-F2144848BB51}&page-id={7D7D0CC0-63BF-4DFB-9373-C732732D1FA0}&object-id={F1E24A57-5CE6-40A2-8442-D349A987FB90}&F&base-path=https://d.docs.live.net/04abe269f00f36a1/文档/数据挖掘/NLP比赛.one)*)*

#用word2vec把单词表示成一个向量，然后用TruncatedSVD进行降维，把原先规模为(文本数，词汇数)的特征#矩阵化为规模为(文本数，主题数)的新特征矩阵

from sklearn.decomposition import TruncatedSVD

#### *from sklearn.feature\_extraction.text import TfidfVectorizer (*[*页首*](onenote:#各种花里胡哨的NL编码&section-id={96A5A6B9-06D6-4B2B-884F-F2144848BB51}&page-id={7D7D0CC0-63BF-4DFB-9373-C732732D1FA0}&object-id={F1E24A57-5CE6-40A2-8442-D349A987FB90}&F&base-path=https://d.docs.live.net/04abe269f00f36a1/文档/数据挖掘/NLP比赛.one)*)*

def tfidf(input\_values, output\_num, output\_prefix, seed=1024):

tfidf\_enc = TfidfVectorizer()

tfidf\_vec = tfidf\_enc.fit\_transform(input\_values) **#只能是值是'a b a c d d xsf xf xsd'的Series或者是list等迭代器**

svd\_tmp = TruncatedSVD(n\_components=output\_num, n\_iter=20, random\_state=seed)

svd\_tmp = svd\_tmp.fit\_transform(tfidf\_vec)

svd\_tmp = pd.DataFrame(svd\_tmp)

svd\_tmp.columns = ['{}\_tfidf\_{}'.format(output\_prefix, i) for i in range(output\_num)]

return svd\_tmp

* + *拓展 如何得到****值是'a b a c d d xsf xf xsd'的Series*** *(*[*页首*](onenote:#各种花里胡哨的NL编码&section-id={96A5A6B9-06D6-4B2B-884F-F2144848BB51}&page-id={7D7D0CC0-63BF-4DFB-9373-C732732D1FA0}&object-id={F1E24A57-5CE6-40A2-8442-D349A987FB90}&F&base-path=https://d.docs.live.net/04abe269f00f36a1/文档/数据挖掘/NLP比赛.one)*)*

y_diff 
NaN 
o 
2 
3 
4 
5 
6 
7 
8 
9 
10 
ship 
6070 
6070 
6070 
6070 
6070 
6070 
6070 
6070 
6070 
6070 
6070 
58.811144 
58.819888 
58.828102 
58 836843 
58.845519 
58.853728 
58.861935 
58.870079 
58.878749 
58.887418 
58.895619 
28.663437 
28.660463 
28.657874 
28.652818 
28.650230 
28.647643 
28.645943 
28.643861 
28.641778 
28.639192 
3.78 
4.32 
4.32 
3.99 
3.99 
389 
3.99 
3.99 
4.32 
3.72 
d 
257 
254 
257 
255 
253 
250 
244 
234 
248 
242 
time 
2019-11-23 2366:og 
20W11-23 2346:06 
2019-11-23233606 
201W11-23 231604 
201W11-23 230604 
2019-11-23 22.5602 
2019-11-23 224559 
2019-11-23 223559 
2019-11-232225:59 
2019-11-23 2215:55 
type 
2.0 
20 
20 
20 
20 
20 
2.0 
2.0 
2.0 
20 
x_diff 
NaN 
417262076 1323422133 
418.922133 1221.910946 
528.439447 
419.395468 
419.548606 
530.325809 
529.050020 
529.206655 
420.153544 
1222.908772 
1221691862 
1221622681 
1121241593 
1222.579325 
1222491249 
1221 _319814 
v_diff 
NaN 
0_54 
-0.33 
o_oo 
-0_10 
0.10 
o_oo 
0.33 
-060 
diff_minutes 
100 
100 
100 
100 
100 
100 
100 
100 
9.0 
anchor 
O 
O 
O 
o 
o 
O 
O 
O 
O 
O 
O 
tm 1 g460 
tm1g43r 
tm 1 g49j 
tm1g4b7 
trn1g501 
tm Ifgpx 
tmlfgrj 
tm Ifgwg 
tm Ifgy6 
tm1funO 
tmlfujw 

tmp = df.groupby('ship')['y\_x'].agg(list).reset\_index()

tmp

o 
2 
3 
4 
ship 
O 
2 
3 
4 
[tml 7m8d, tml 7m98, tm17m6q, tml 7m7k, tm17mk4, 
[tmOgnhx, tmOgn7u, tmOgncu, tm()gpll , 
[tmlsryw, tml sryw, tmlsryw, tmlsryw, tmlsryw, 
(Ohngnu, tjhngnu, tjhA9ns, tjhngns, tjhngns, 
ltmpwkp5, tmpwkp5, tmpwkp5, tmpwkp5, tmpwkp5, 

tmp['join'] = tmp['y\_x'].apply(lambda x: ' '.join(x))

tmp

o 
2 
3 
4 
Ship 
O 
2 
3 
4 
[tml 7m8d, tm17m98, tml 7m6q, tm17m7k, tm17mk4, 
[tmOgnhx, tmOgn7u, tmOgncu, tmOgp11 , 
(tmlsryw, tmlsryw, tmlsmv, tmlsryw, tmlsryw, 
[tjhn9nu, tjhn9nu, tjhn9ns, tjhn9ns, tjhn9ns, 
[tmpwkp5, tmpwkp5, tmpwkp5, tmp%'kp5, tmpw•kp5, 
Join 
tml 7m8d tm17m98 tm17m6q tm17m7k tml 7mk4 tm17mm. 
tmOgnhx tmOgn7u tmOgndx tmOgncu tmOgp11 tmOgpO 
tmlsryw tm Isryw tmlsryw' tmlsryw tmlsryw tm Isry. 
tjhn9nu tjhn9nu tjhn9ns tjhn9ns tjhn9ns tjhn9n, 
tmpwkp5 tmpwkp5 tmpwkp5 tmpwkp5 tmpw•kp5 tmpwkp 

## Word2Vec ([页首](onenote:#各种花里胡哨的NL编码&section-id={96A5A6B9-06D6-4B2B-884F-F2144848BB51}&page-id={7D7D0CC0-63BF-4DFB-9373-C732732D1FA0}&object-id={F1E24A57-5CE6-40A2-8442-D349A987FB90}&F&base-path=https://d.docs.live.net/04abe269f00f36a1/文档/数据挖掘/NLP比赛.one))

* + *资料 (*[*页首*](onenote:#各种花里胡哨的NL编码&section-id={96A5A6B9-06D6-4B2B-884F-F2144848BB51}&page-id={7D7D0CC0-63BF-4DFB-9373-C732732D1FA0}&object-id={F1E24A57-5CE6-40A2-8442-D349A987FB90}&F&base-path=https://d.docs.live.net/04abe269f00f36a1/文档/数据挖掘/NLP比赛.one)*)*

[gensim训练word2vec及相关函数与功能理解](https://blog.csdn.net/sinat_26917383/article/details/69803018)

[Word2Vec的参数解释及其实战](https://www.jianshu.com/p/972d0db609f2)

[word2vec进阶之skim-gram和CBOW模型（Hierarchical Softmax、Negative Sampling）](https://blog.csdn.net/u014665013/article/details/79622400)

[Embedding和Word2Vec实战](https://www.cnblogs.com/dogecheng/p/11565530.html)

[Tensorflow官方文档word2vec\_basic.py中文注释](https://www.cnblogs.com/Yorkisme/p/7618002.html)

[word2vec之tensorflow（skip-gram）实现](https://my.oschina.net/u/4393418/blog/3251575/print)

[本地视频](file:///D:/机器学习与人工智能/机器学习&深度学习/就业班/31%20深度学习必备原理与实战2/算法：自然语言处理-word2vec)

* + *定义 (*[*页首*](onenote:#各种花里胡哨的NL编码&section-id={96A5A6B9-06D6-4B2B-884F-F2144848BB51}&page-id={7D7D0CC0-63BF-4DFB-9373-C732732D1FA0}&object-id={F1E24A57-5CE6-40A2-8442-D349A987FB90}&F&base-path=https://d.docs.live.net/04abe269f00f36a1/文档/数据挖掘/NLP比赛.one)*)*
    - from gensim.models import word2vec#把每一个单词表示成一个向量。来计算文档相似度计算等等。
    - NLP 里的词语，是人类的抽象总结，是符号形式的（比如中文、英文、拉丁文等等），所以需要把他们转换成数值形式，或者说——嵌入到一个数学空间里，这种嵌入方式，就叫词嵌入（word embedding)，而 Word2vec，就是词嵌入（ word embedding) 的一种
    - 主要用途

用于词向量的获取。即根据“上下文-单词”矩阵进行学习，得到融入了上下文共现特征的词向量。

* + 优缺点

[1] 优点：由于Word2vec会考虑上下文，跟之前的Embedding方法相比，效果要更好；比之前的Embedding方法维度更少，所以速度更快；通用性很强，可以用在各种 NLP 任务中。

[2] 缺点：由于词和向量是一对一的关系，所以多义词的问题无法解决；此外，Word2vec是一种静态的方式，虽然通用性强，但是无法针对特定任务做动态优化。

* + *参数 (*[*页首*](onenote:#各种花里胡哨的NL编码&section-id={96A5A6B9-06D6-4B2B-884F-F2144848BB51}&page-id={7D7D0CC0-63BF-4DFB-9373-C732732D1FA0}&object-id={F1E24A57-5CE6-40A2-8442-D349A987FB90}&F&base-path=https://d.docs.live.net/04abe269f00f36a1/文档/数据挖掘/NLP比赛.one)*)*

· sentences：可以是一个list，对于大语料集，建议使用BrownCorpus,Text8Corpus或lineSentence构建。

· size：是指特征向量的维度，默认为100。

· alpha: 是初始的学习速率，在训练过程中会线性地递减到min\_alpha。

· window：窗口大小，表示当前词与预测词在一个句子中的最大距离是多少。

· min\_count: 可以对字典做截断. 词频少于min\_count次数的单词会被丢弃掉, 默认值为5。

· seed：用于随机数发生器。与初始化词向量有关。

· workers：用于控制训练的并行数。

· min\_alpha：学习率的最小值。

· sg： 用于设置训练算法，默认为0，对应CBOW算法（速度快，准确度低）；sg=1则采用skip-gram算法（速度慢准确度高），[**请参考这里**](https://www.cnblogs.com/june0507/p/9412989.html)

· hs: 如果为1则会采用hierarchica softmax技巧(对低频词的效果更好)。如果设置为0（默认），则使用negative sampling（负采样）(对高频词的效果更好)。

· hashfxn： hash函数来初始化权重，默认使用python的hash函数。

· iter： 迭代次数，默认为5。

* + *入门使用 (*[*页首*](onenote:#各种花里胡哨的NL编码&section-id={96A5A6B9-06D6-4B2B-884F-F2144848BB51}&page-id={7D7D0CC0-63BF-4DFB-9373-C732732D1FA0}&object-id={F1E24A57-5CE6-40A2-8442-D349A987FB90}&F&base-path=https://d.docs.live.net/04abe269f00f36a1/文档/数据挖掘/NLP比赛.one)*)*

# 初始化 ([页首](onenote:#各种花里胡哨的NL编码&section-id={96A5A6B9-06D6-4B2B-884F-F2144848BB51}&page-id={7D7D0CC0-63BF-4DFB-9373-C732732D1FA0}&object-id={F1E24A57-5CE6-40A2-8442-D349A987FB90}&F&base-path=https://d.docs.live.net/04abe269f00f36a1/文档/数据挖掘/NLP比赛.one))

from gensim.models import word2vec

sentences = [['first', 'sentence'], ['second', 'sentence','is']]

# 训练 ([页首](onenote:#各种花里胡哨的NL编码&section-id={96A5A6B9-06D6-4B2B-884F-F2144848BB51}&page-id={7D7D0CC0-63BF-4DFB-9373-C732732D1FA0}&object-id={F1E24A57-5CE6-40A2-8442-D349A987FB90}&F&base-path=https://d.docs.live.net/04abe269f00f36a1/文档/数据挖掘/NLP比赛.one))

model=word2vec.Word2Vec(sentences,min\_count=5,size=50)

# min\_count,频数阈值，大于等于1的保留，默认值为5；

# size是神经网络的隐藏层单元数，默认为100；

# workers=4，default = 1 worker = no parallelization 只有在机器已安装 Cython 情况下才会起到作用。

#结果查看 ([页首](onenote:#各种花里胡哨的NL编码&section-id={96A5A6B9-06D6-4B2B-884F-F2144848BB51}&page-id={7D7D0CC0-63BF-4DFB-9373-C732732D1FA0}&object-id={F1E24A57-5CE6-40A2-8442-D349A987FB90}&F&base-path=https://d.docs.live.net/04abe269f00f36a1/文档/数据挖掘/NLP比赛.one))

model.similarity("好", "还行")#计算两个词之间的余弦距离

计算机生成了可选文字:
e. 642981583608 

model.most\_similar("滋润", topn=10)#计算余弦距离最接近“滋润”的10个词

讠 跆 显 
高 0 
名 服 
补 水 
清 爽 
真 0 〕 
0 ， 995M71520d2 
9 。 98519398457 彐 
· 978088299 的 2 
9 g6g187 刁 969 
9 。 957649161816 
0 ， 9695 丿 0812225 
9 。 958645284176 
0 ， 928M3763 ％ 5 
9 91177d456591 
0 。 90943989217 

#存储和加载模型 ([页首](onenote:#各种花里胡哨的NL编码&section-id={96A5A6B9-06D6-4B2B-884F-F2144848BB51}&page-id={7D7D0CC0-63BF-4DFB-9373-C732732D1FA0}&object-id={F1E24A57-5CE6-40A2-8442-D349A987FB90}&F&base-path=https://d.docs.live.net/04abe269f00f36a1/文档/数据挖掘/NLP比赛.one))

model.save('/model/word2vec\_model')

new\_model=gensim.models.Word2Vec.load('/model/word2vec\_model')

#获取词向量 ([页首](onenote:#各种花里胡哨的NL编码&section-id={96A5A6B9-06D6-4B2B-884F-F2144848BB51}&page-id={7D7D0CC0-63BF-4DFB-9373-C732732D1FA0}&object-id={F1E24A57-5CE6-40A2-8442-D349A987FB90}&F&base-path=https://d.docs.live.net/04abe269f00f36a1/文档/数据挖掘/NLP比赛.one))

model['computer']

# 寻找不合群的词 ([页首](onenote:#各种花里胡哨的NL编码&section-id={96A5A6B9-06D6-4B2B-884F-F2144848BB51}&page-id={7D7D0CC0-63BF-4DFB-9373-C732732D1FA0}&object-id={F1E24A57-5CE6-40A2-8442-D349A987FB90}&F&base-path=https://d.docs.live.net/04abe269f00f36a1/文档/数据挖掘/NLP比赛.one))

y4 = model.doesnt\_match("书 书籍 教材 很".split())

#### *拓展 (*[*页首*](onenote:#各种花里胡哨的NL编码&section-id={96A5A6B9-06D6-4B2B-884F-F2144848BB51}&page-id={7D7D0CC0-63BF-4DFB-9373-C732732D1FA0}&object-id={F1E24A57-5CE6-40A2-8442-D349A987FB90}&F&base-path=https://d.docs.live.net/04abe269f00f36a1/文档/数据挖掘/NLP比赛.one)*)*

#sentences格式 ([页首](onenote:#各种花里胡哨的NL编码&section-id={96A5A6B9-06D6-4B2B-884F-F2144848BB51}&page-id={7D7D0CC0-63BF-4DFB-9373-C732732D1FA0}&object-id={F1E24A57-5CE6-40A2-8442-D349A987FB90}&F&base-path=https://d.docs.live.net/04abe269f00f36a1/文档/数据挖掘/NLP比赛.one))

pd.Series(['1 2 3','2 3 54']

['1','2','3'],['2','3','54']

pd.Series([['1','2','3'],['2','3','54']]

* + *举例 与TruncatedSVD结合实现LSI/LSA（隐含语义分析） (*[*页首*](onenote:#各种花里胡哨的NL编码&section-id={96A5A6B9-06D6-4B2B-884F-F2144848BB51}&page-id={7D7D0CC0-63BF-4DFB-9373-C732732D1FA0}&object-id={F1E24A57-5CE6-40A2-8442-D349A987FB90}&F&base-path=https://d.docs.live.net/04abe269f00f36a1/文档/数据挖掘/NLP比赛.one)*)*

#用word2vec把单词表示成一个向量，然后用TruncatedSVD进行降维，把原先规模为(文本数，词汇数)的特征#矩阵化为规模为(文本数，主题数)的新特征矩阵

def hashfxn(astring):

return ord(astring[0])[#返回字符串的ASCII 数值，或者 Unicode 数值](https://www.runoob.com/python/python-func-ord.html)

from sklearn.decomposition import TruncatedSVD

from gensim.models import word2vec

def w2v\_feat(df, group\_id, feat, length):

print('start word2vec ...')

data\_frame = df.groupby(group\_id)[feat].agg(list).reset\_index()

model = Word2Vec(data\_frame[feat].values, size=length, window=5, min\_count=1, sg=1, hs=1,

workers=1, iter=10, seed=1, hashfxn=hashfxn)

data\_frame[feat] = data\_frame[feat].apply(lambda x: pd.DataFrame([model[c] for c in x]))

for m in range(length):

data\_frame['w2v\_{}\_mean'.format(m)] = data\_frame[feat].apply(lambda x: x[m].mean())

del data\_frame[feat]

return data\_frame

* + *拓展 (*[*页首*](onenote:#各种花里胡哨的NL编码&section-id={96A5A6B9-06D6-4B2B-884F-F2144848BB51}&page-id={7D7D0CC0-63BF-4DFB-9373-C732732D1FA0}&object-id={F1E24A57-5CE6-40A2-8442-D349A987FB90}&F&base-path=https://d.docs.live.net/04abe269f00f36a1/文档/数据挖掘/NLP比赛.one)*)*

**1）CBoW vs Skip-Gram**

* + CBoW更快一些。CBoW对于高频词效果较好，低频词常受到较少注意。窗口大小常5左右。
  + Skip-Gram更慢一些。Skip-Gram对于低频词效果更好，小数据下表现依旧好。窗口大小常10左右。
  + 如果是用一个词语作为输入，来预测它周围的上下文，那这个模型叫做『Skip-gram 模型』
  + 而如果是拿一个词语的上下文作为输入，来预测这个词语本身，则是 『CBOW 模型』

对于"Yesterday was really a \_\_\_\_ day."：

CBoW认为最可能是beautiful/nice，delightful受到较少注意；Skip-Gram则不会将delightful与beautiful/nice比较，而是作为一组新观测值。

**2）Hierarchical Softmax vs Negative Sampling**

* + Hierarchical Softmax  
    优点是对低频词的效果更好。因为表示低频词的叶子节点会不可避免地继承祖先节点的向量表示，这个祖先节点可能会受到其他高频词的影响。  
    缺点是如果所需要的输出词很生僻，得一直往下走很久。
  + Negative Sampling  
    对高频词效果更好。向量维度较低时效果更好，维度高时近似误差会比较大。

## [Countvectorizer](https://scikit-learn.org/0.18/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html#sklearn-feature-extraction-text-countvectorizer) ([页首](onenote:#各种花里胡哨的NL编码&section-id={96A5A6B9-06D6-4B2B-884F-F2144848BB51}&page-id={7D7D0CC0-63BF-4DFB-9373-C732732D1FA0}&object-id={F1E24A57-5CE6-40A2-8442-D349A987FB90}&F&base-path=https://d.docs.live.net/04abe269f00f36a1/文档/数据挖掘/NLP比赛.one))

## 综合案例一 已知船只以米为单位的直角坐标 ([页首](onenote:#各种花里胡哨的NL编码&section-id={96A5A6B9-06D6-4B2B-884F-F2144848BB51}&page-id={7D7D0CC0-63BF-4DFB-9373-C732732D1FA0}&object-id={F1E24A57-5CE6-40A2-8442-D349A987FB90}&F&base-path=https://d.docs.live.net/04abe269f00f36a1/文档/数据挖掘/NLP比赛.one))