利用Word2Vec判断文言、白话文

数据分析挖掘与算法 2019-04-24

4个小节，预计用时**30分钟**。

请打开您的电脑，按照步骤一步步完成哦！

本教程基于**Python 3.5**。

原创者：**SofaSofa TeamM** | 修改校对：SofaSofa TeamC |

0. 前言

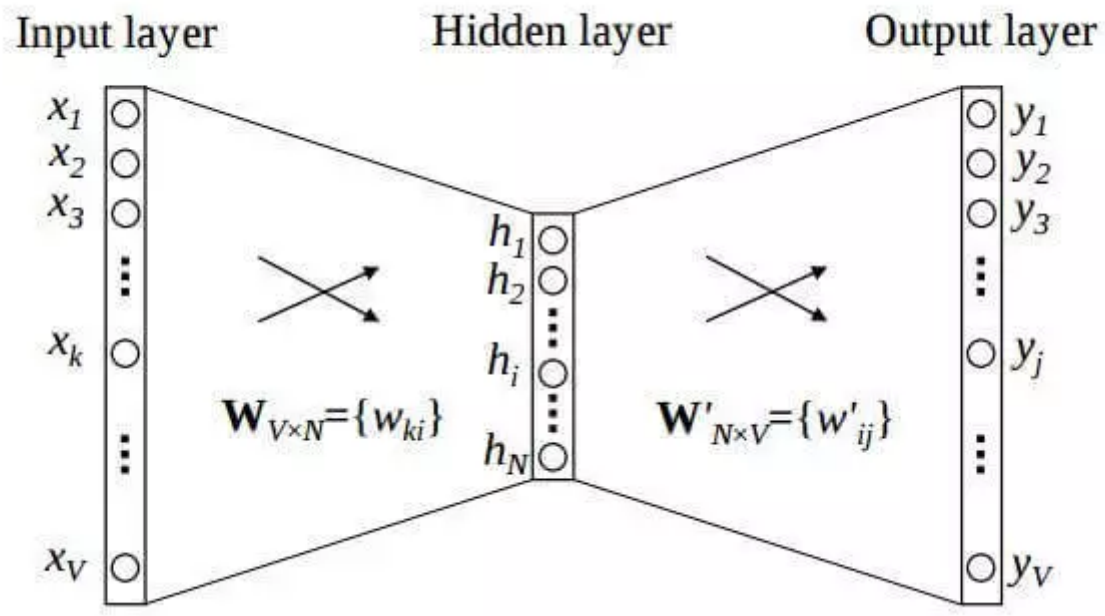
我们将用**最最简短**的语言告诉大家word2vec是什么，并且使用gensim来获得词向量。最后利用词向量来判断一个语句是文言文还是白话文，也就是完成“**机器读中文2：辨古识今**”。

1. 什么是Word2Vec?

Word2Vec是一种自然语言处理的模型，简单说来，就是把一个word表示为一个vector，所以叫Word2Vec。对于中文，一个Word可以是一个单词，也可以是一个词。

把词表示为向量，这样做有什么好处呢？第一是可以降维，如果把每个词都做one-hot编码，那么维度会特别大，如果把每个词都表示为长度为nn的向量，就可以完全省去one-hot带来的高维度。第二点，是可以挖掘词与词的联系。如果训练样本够大，我们会发现向量“猫”和向量“喵”会更相似，而不是“狗”。在英文中一个经典例子是，向量“Queen”-向量“Female”=向量“King”-向量“Male”。

Word2Vec模型是一个只有一个隐藏层的神经网络模型（如下）。每个词是一条数据，目的是预测这个词的后一个（或者kk个词）。输入数据是one-hot的向量，输出层的结果是预测该词后面出现的词。那么Word2Vec中的vector在哪里呢？这个向量其实就是隐藏层的数值。



2. 利用gensim计算Word2Vec

gensim 是python中自然语言处理非常强大的包，而且极易使用。首先，我们在python中引用该包。

```
from gensim.models import Word2Vec
```

下面我们来读取“机器读中文2：辨古识今”中的数据集。

```
import pandas as pd
train = pd.read_csv('train.txt')
test = pd.read_csv('test.txt')
texts = list(train['text']) + list(test['text'])
```

不妨先看看 texts 里的内容有哪些。

```
print(texts[:10])
```

```
['来扰乱天子的边防', '崤山以东地区虽然混乱', '秦昭王闻之 使人遗赵王书 愿以十五城请易璧', '臣不胜受恩感激', '现在如果把东西寄存在别人处', '南北朝所以不治 文采胜质厚也', '而流贼进攻更急 城里有许多不同的意见', '吾上有三兄 皆不幸早世', '没有人不讲究熏衣剃面', '王曰 善']
```

接下来，就可以训练Word2Vec模型了。这里我们把每一个字作为一个word，每个字都会被表示为一个向量。一键完成，非常方便。

```
ndims = 50
model = Word2Vec(sentences=texts, size=ndims, window=5)
```

模型中 `sentences` 是训练素材， `size` 是指vector的长度， `window` 是指窗长，也就是预测该词之后的5个词。这样，每个字都被表示成了一个长度为50的向量。下面我们不妨试几个词。

```
print(model.wv['之'])
```

```
[-0.02306273  0.7689658  -0.23238362 -0.36329043  0.14156663  0.29761788
 -0.45338956 -0.7482197  -0.06813935 -0.4105922  -0.3896525  -0.5217305
  0.11865879 -0.32430476 -0.53716886  0.25903523  0.11261824 -0.3138387
 -0.46736085 -0.04374949 -0.2901372   0.04580646 -0.24377792  0.36069182
  0.45043638 -0.82071906 -0.77037716  0.65637636  0.45035675 -0.05211991
  0.6374315  -0.52816385  0.09859632 -0.41244134  0.7426563   0.01779771
  0.10559028 -0.04655827  0.26579723 -0.48558313  0.38831276 -0.3439704
  0.26713774 -0.267362   -0.22380458  0.99069804 -0.28053552  0.16005835
  0.05433292  0.40896153]
```

```
print(model.wv['的'])
```

```
[-5.1170345e-02  7.3522794e-01 -2.1519105e-01 -3.3946022e-01
  1.1915052e-01  3.1610066e-01 -4.1824239e-01 -7.6163346e-01
 -4.0698811e-02 -4.0355769e-01 -3.0365336e-01 -4.8146579e-01
  1.3719502e-01 -3.8396400e-01 -4.9982443e-01  2.2536282e-01
  8.1103407e-02 -2.8362677e-01 -4.9123862e-01  7.8462285e-04
 -3.2690132e-01  6.6004813e-02 -2.5450531e-01  3.4221938e-01
  4.4872758e-01 -7.7068639e-01 -7.8395563e-01  6.9241500e-01
  5.3227925e-01 -3.4683388e-02  6.6531086e-01 -5.2474988e-01
  1.8498762e-01 -4.4998875e-01  7.4666858e-01  7.6578781e-02
  1.6198398e-01 -8.5742459e-02  2.8727862e-01 -4.8327830e-01
  3.2108101e-01 -3.9720288e-01  2.3086597e-01 -3.2169360e-01
 -2.2850281e-01  1.0791894e+00 -2.7565178e-01  1.5590596e-01
 -3.6102833e-04  4.3195924e-01]
```

那么，哪些字和“之”最相近呢？答案如下，果不其然，最接近的也都是文言文中的常用字。

```
print(model.wv.most_similar('之', topn=5))
```

```
[('其', 0.9995831847190857), ('者', 0.9995520114898682), ('以', 0.9995492696762085), ('此', 0.9995350241661072), ('欲', 0.9994945526123047)]
```

再看看“的”，最接近“的”的字也都是白话文中的常用字。

```
print(model.wv.most_similar('的', topn=5))
```

```
[('了', 0.9994625449180603), ('就', 0.9994519352912903), ('这', 0.9994337558746338), ('个', 0.999407172203064), ('都', 0.9993969798088074)]
```

3. 利用词向量进行数据可视化

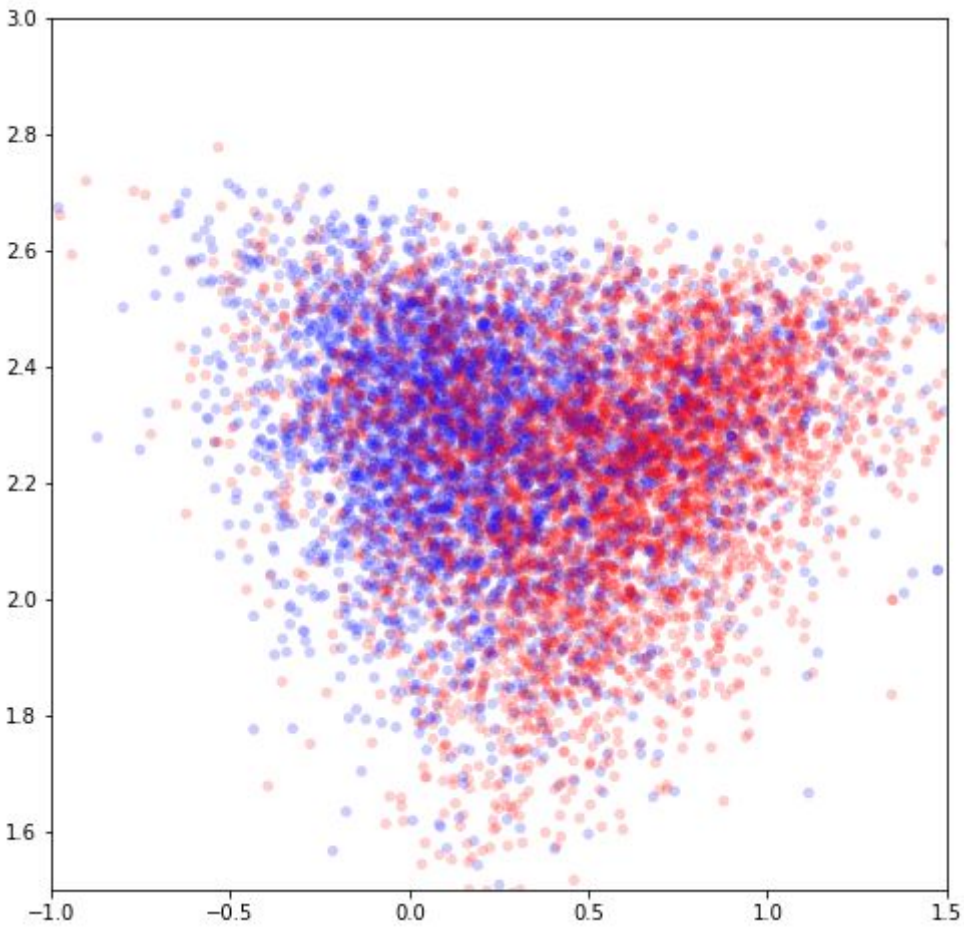
我们可以利用Word2Vec对文本进行数据可视化处理。首先，我们把每个字表示为长度为2的向量（也就是 `ndims=2` ），然后对每一句话中所有字的向量求均值。这样每句话都会被表示成平面上的一个点。

```
ndims = 2
model = Word2Vec(sentences=texts, size=ndims, window=5)

total = len(texts)
vecs = np.zeros([total, ndims])for i, sentence in enumerate(texts):
    counts, row = 0, 0
    for char in sentence:
        try:
            if char != ' ':
                row += model.wv[char]
                counts += 1
        except:
            pass
    if counts == 0:
        print(sentence)
    vecs[i, :] = row / counts
```

根据上面代码，我们得到了训练集 `train` 的向量表达 `vecs` 。 `vecs` 中每一行代表一句话，用一个长度为2的向量表示。结合训练集中的真实标签 `train['y']` ，我们可以在坐标系中绘出词向量。

```
import matplotlib.pyplot as plt
plt.figure(figsize=(8, 8))
plt.axis([-1, 1.5, 1.5, 3])
colors = list(map(lambda x: 'red' if x == 1 else 'blue', train['y']))
plt.scatter(vecs[:, 0], vecs[:, 1], c=colors, alpha=0.2, s=30, lw=0)print('Word2Vec: 白话文(蓝色)与文言文(红色)')
plt.show()
```



尽管只用了长度为2的向量，文言文（红色）和白话文（蓝色）已经可以较好区分。

4. 利用Word2Vec建立文言、白话分类器

首先，引用包、读取数据。

```
import pandas as pd
import numpy as np
from sklearn.tree import DecisionTreeClassifierfrom gensim.models import Word2V
ec

train = pd.read_csv('train.txt')
test = pd.read_csv('test.txt')
submit = pd.read_csv('sample_submit.csv')
```

获取texts

```
total = len(train) + len(test)
n_train = len(train)

labeled_texts = []

texts = list(train['text']) + list(test['text'])
```

利用Word2Vec，将每句话转成长度为100的数值向量。

```
ndims = 100
model = Word2Vec(sentences=texts, size=ndims)

vecs = np.zeros([total, ndims])for i, sentence in enumerate(texts):
    counts, row = 0, 0
    for char in sentence:
        try:
            if char != ' ':
                row += model.wv[char]
                counts += 1
        except:
            pass
    if counts == 0:
        print(sentence)
    vecs[i, :] = row / counts
```

利用sklearn中的DecisionTreeClassifier，建立决策树分类模型并得到最终的分类结果，保存至 `my_prediction.csv` 。

```
clf = DecisionTreeClassifier(max_depth=3, random_state=100)
clf.fit(vecs[:n_train], train['y'])
submit['y'] = clf.predict_proba(vecs[n_train:])[:, 1]
submit.to_csv('my_prediction.csv', index=False)
```

我们不妨看看，我们的预测效果如何。（为了方便，我们把预测值大于0.5的标为1文言文，小于0.5的标为0白话文。）

```
test['pred'] = (submit['y'] > 0.5).astype(int)
test.head(20)
```

	id	text	pred
0	5000	后来又合并为七个强国	0
1	5001	与吾父居者 今其室十无二三焉	1
2	5002	那么政局就会安定了	0
3	5003	我资米若薪于百姓 后之人必尔乎索之	1
4	5004	昔虞国宫之奇少长於君 君狎之	1
5	5005	我们怎能学他那种穷途的哭泣	0
6	5006	迫而视之 乃前寄辞者	1
7	5007	但后来执掌法律的官员却以耗费国家资财的罪名上书弹劾	0

	id	text	pred
8	5008	你们因为张公的智慧得到了生存 他就是你们的再生父母	0
9	5009	以父荫补弘文校书郎 擢累谏议大夫	1
10	5010	操蛇之神闻之 惧其不已也 告之于帝	1
11	5011	夫不通礼义之旨 至于君不君	1
12	5012	今老矣 无能为也已	1
13	5013	这难道不是得到它的好处却背叛它的恩惠吗	0
14	5014	念诸父与诸兄 皆康强而早世	1
15	5015	晋侯 秦伯围郑 以其无礼于晋 且贰于楚也	1
16	5016	陈王闻 乃使武平君畔为将军 监郯下军	1
17	5017	汗流浃背 吃不下东西 说	0
18	5018	业已整齐门内 提撕子孙	1
19	5019	只要有胜过我的地方 就很可贵	0

这个分类结果还是比较让人满意的。上传到排行榜看看结果如何吧！

整个分类过程的完整代码如下：

```
# -*- coding: utf-8 -*-import pandas as pd
import numpy as np
from sklearn.tree import DecisionTreeClassifierfrom gensim.models import Word2V
ec

train = pd.read_csv('train.txt')
test = pd.read_csv('test.txt')
submit = pd.read_csv('sample_submit.csv')

total = len(train) + len(test)
n_train = len(train)

labeled_texts = []

texts = list(train['text']) + list(test['text'])

ndims = 100
model = Word2Vec(sentences=texts, size=ndims)

vecs = np.zeros([total, ndims])for i, sentence in enumerate(texts):
    counts, row = 0, 0
```

```
    for char in sentence:
        try:
            if char != ' ':
                row += model.wv[char]
                counts += 1
        except:
            pass
    if counts == 0:
        print(sentence)
    vecs[i, :] = row / counts

clf = DecisionTreeClassifier(max_depth=3, random_state=100)
clf.fit(vecs[:n_train], train['y'])
submit['y'] = clf.predict_proba(vecs[n_train:])[:, 1]
submit.to_csv('my_prediction.csv', index=False)
```

猜你可能喜欢

基于最小二乘法的线性回归拟合

多元线性回归、逐步回归、逻辑回归的总结

基于梯度下降法的——线性回归拟合

Python系列之——好用的Python开发工具

阅读原文