电子商务数据分析

第6.2章 编码与词向量

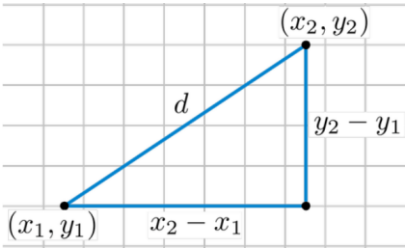
朱桂祥 (9120201070@nufe.edu.cn)
南京财经大学信息工程学院
江苏省电子商务重点实验室
电子商务信息处理国家级国际联合研究中心
电子商务交易技术国家地方联合工程实验室

校训是什么？
西南联大：刚毅坚卓 精神永驻

中国抗日战争开始后高校内迁设于昆明的一所综合性大学。1937年11月1日，由国立北京大学、国立清华大学、私立南开大学在长沙组建成立的国立长沙临时大学在长沙开学（这一天也成为西南联大校庆日）。由于长沙连遭日机轰炸，1938年2月中旬，经中华民国教育部批准，长沙临时大学分三路西迁昆明。1938年4月，改称国立西南联合大学。西南联大前后共存在了8年零11个月，“内树学术自由之规模，外来民主堡垒之称号”，保存了抗战时期的重要科研力量，培养了一大批卓有成就的优秀人才，为中国和世界的发展进步作出了杰出贡献。

https://tv.cctv.com/2014/10/30/VIDE1414633443593241.shtml

One-Hot编码



One-Hot编码定义：

One-Hot编码（独热编码），又称为一位有效编码，主要是采用***N***位状态寄存器来对***N*个状态**进行编码，每个状态都由他独立的寄存器位，并且在任意时候只有一位有效。

One-Hot编码是**分类变量**作为**二进制向量**的表示。这首先要求将分类值映射到整数值。然后，每个整数值被表示为二进制向量，除了整数的索引之外，它都是零值，它被标记为1。

为什么要用One-Hot编码？

使用独热编码（One-Hot Encoding），将离散特征的取值扩展到了欧式空间，离散特征的某个取值就对应欧式空间的某个点。将离散型特征使用独热编码（One-Hot Encoding），会让特征之间的距离计算更加合理。

演员--0；厨师--1；公务员--2；工程师--3；律师--4。

两个工作之间的欧式距离是：

Euclidean Distance(演员，厨师)=1; Euclidean Distance(演员，公务员)=2;

Euclidean Distance(演员，工程师） = 3

显然这样的表示，计算出来的特征的距离是不合理。那如果使用独热编码（One-Hot Encoding），则得到d(演员，厨师) = 1与d(演员，公务员）都是1。那么，两个工作之间的距离就都是sqrt(2)。即每两个工作之间的距离是一样的，显得更合理。

One-Hot编码

什么时候不需要使用独热编码？

1.离散特征的取值之间没有大小意义时，可以使用独热编码。

当离散特征的取值之间有大小意义或者有序时，比如衣服尺寸: [X, XL, XXL]，那么就不能使用独热编码，而使用数值的映射{X: 1, XL: 2, XXL: 3}。

2.如果特征是离散的，并且不用独热编码就可以很合理的计算出距离，就没必要进行独热编码。

3.有些并不是基于向量空间度量的算法，数值只是类别符号，没有偏序关系，就不用进行独热编码。

4.如果原本的标签编码是有序的，就不必独热编码了，因为会丢失顺序信息。

One-Hot编码的缺陷

（1）忽略了真实数据之间的相关性

onehot编码的一个缺陷就是，它使得标签中的所有类别彼此之间距离或相似度都是一致的，因此，在自然语言处理中，我们还要把单词做一下word2vec做一下编码，使得语义相近的词，他们的编码向量也比较一致。在视觉模型分类中，一般使用onehot即可，因为我们的类别数不会像单词那么多。

（2）类别数量很多时，onehot向量会过长

当类别的数量很多时，特征空间会变得非常大，成为一个高维稀疏矩阵。在这种情况下，一般可以用PCA来减少维度。而且one hot encoding+PCA这种组合在实际中也非常有用。

One-Hot编码

对["中国", "美国", "日本"]进行one-hot编码，
怎么做呢？
1.确定要**编码的对象--["中国", "美国", "日本", "美国"]**，
2.确定**分类变量--中国　美国　日本**，共3种类别；
3.以上问题就相当于，**有4个样本，每个样本有3个特征**，将其转化为**二进制向量**表示，我们首先进行**特征的整数编码**：中国--0，美国--1，日本--2，并将特征按照从小到大排列
得到one-hot编码如下：
["中国", "美国", "日本", "美国"] ---> [[1,0,0], [0,1,0], [0,0,1], [0,1,0]]

	中国	美国	日本
中国	1	0	0
美国	0	1	0
日本	0	0	1
美国	0	1	0

One-Hot编码

One-Hot编码优缺点：

比如我们要对 “hello world” 进行one-hot编码，怎么做呢？

1.确定要**编码的对象：hello world，**

2.确定**分类变量： hello空格world，**共27种类别（26个小写字母 + 空格，）；

3.以上问题就相当于，**有11个样本，每个样本有27个特征，**将其转化为**二进制向量**表示，

这里有一个前提，特征排列的顺序不同，对应的二进制向量亦不同（比如我把空格放在第一列和a放第一列，one-hot编码结果肯定是不同的）

因此我们必须要事先约定特征排列的顺序：

1、27种特征首先进行整数编码：a--0，b--1，c--2，.....，z--25，空格—26。

2、27种特征按照整数编码的大小从前往后排列。

分类变量

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	空
h	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
e	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
l	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
l	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
o	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
空	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
w	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
o	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
r	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
l	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
d	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

二进制向量

One-Hot编码

通过sklearn实现One-Hot编码：

In [13]:
```
from sklearn import preprocessing
encoder = preprocessing.OneHotEncoder()
#4个特征:
#第一个特征（第一列）为[0, 1, 3, 1]
#第一个特征有三类特征值[0, 1, 3]: One-Hot Encoding后采用三个编码: [100, 010, 001]
#同理第二个特征列可将两类特征值[2, 3]表示为[10, 01]
#第三个特征将4类特征值[1, 4, 5, 7]表示为[1000, 0100, 0010, 0001]
#第四个特征将3类特征值[1, 3, 5]表示为[100, 010, 001]
encoder.fit([["中国"],["美国"],["日本"],["韩国"]])
encoded_vector = encoder.transform([["中国"]]).toarray()
print("中国的Encoded Vector =", encoded_vector)
encoded_vector = encoder.transform([["美国"]]).toarray()
print("美国的Encoded Vector =", encoded_vector)
encoded_vector = encoder.transform([["中国"],["美国"]]).toarray()
print("中国美国的Encoded Vector =", encoded_vector)
```

```
中国的Encoded Vector = [[1. 0. 0. 0.]]
美国的Encoded Vector = [[0. 0. 1. 0.]]
中国美国的Encoded Vector = [[1. 0. 0. 0.]
 [0. 0. 1. 0.]]
```

One-Hot编码

通过sklearn实现One-Hot编码：

In [8]:
```
from sklearn import preprocessing
encoder = preprocessing.OneHotEncoder()
#4个特征：
#第一个特征（第一列）为[0, 1, 3, 1]
#第一个特征有三类特征值[0, 1, 3]：One-Hot Encoding后采用三个编码：[100, 010, 001]
#同理第二个特征列可将两类特征值[2, 3]表示为[10, 01]
#第三个特征将4类特征值[1, 4, 5, 7]表示为[1000, 0100, 0010, 0001]
#第四个特征将3类特征值[1, 3, 5]表示为[100, 010, 001]
encoder.fit([
    [0, 2, 7, 1],
    [1, 3, 5, 3],
    [3, 3, 1, 5],
    [1, 2, 4, 5]
])
encoded_vector = encoder.transform([[3, 2, 7, 5]]).toarray()
print("One-Hot vector Encoded Vector =", encoded_vector)
#[[0. 0. 1.    1. 0.      0. 0. 0. 1.      0. 0. 1.]]
```

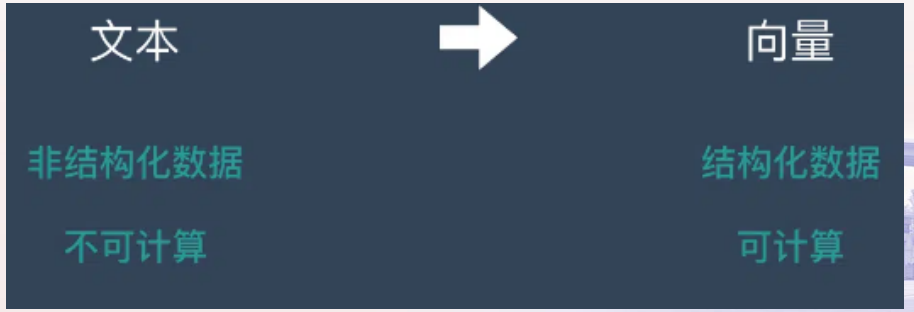
```
One-Hot vector Encoded Vector = [[0. 0. 1. 1. 0. 0. 0. 0. 1. 0. 0. 1.]]
```

Word2Vec编码

Word2Vec是google在2013年推出的一个NLP工具，它的特点是能够将单词转化为向量来表示，这样词与词之间就可以定量的去度量他们之间的关系，挖掘词之间的联系。

如果词的语义相近，它们的词向量在向量空间中也相互接近，这使得词语的向量化建模更加精确，可以改善现有方法并提高鲁棒性。词向量已被证明在许多自然语言处理问题，如：机器翻译，标注问题，实体识别等问题中具有非常重要的作用。

Word2Vec可将“不可计算”“非结构化”的词转化为“可计算”“结构化”的向量



Word2Vec编码

1-of-N Encoding

apple = [1 0 0 0 0]

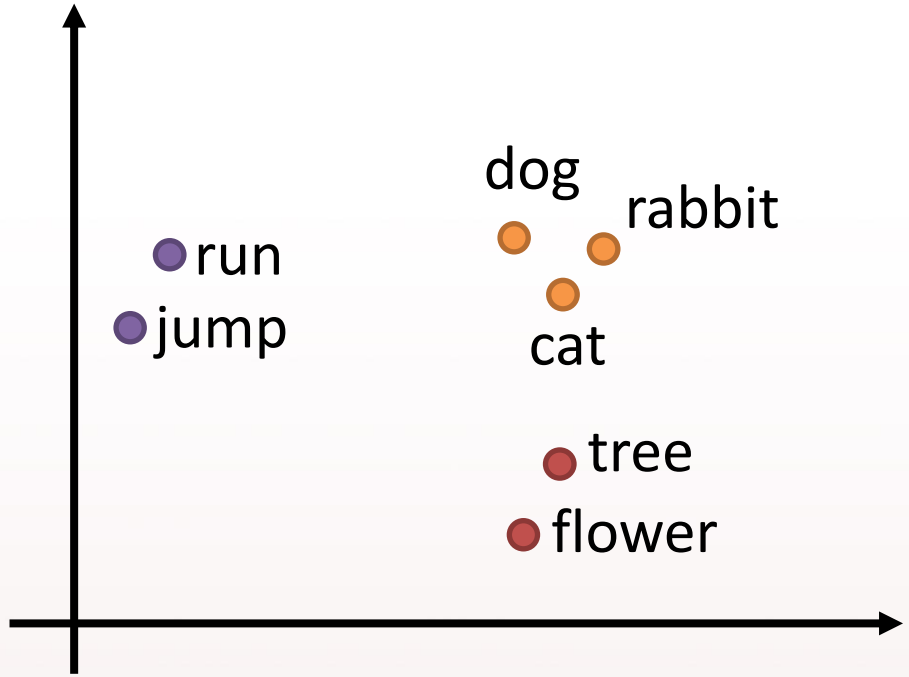
bag = [0 1 0 0 0]

cat = [0 0 1 0 0]

dog = [0 0 0 1 0]

elephant = [0 0 0 0 1]

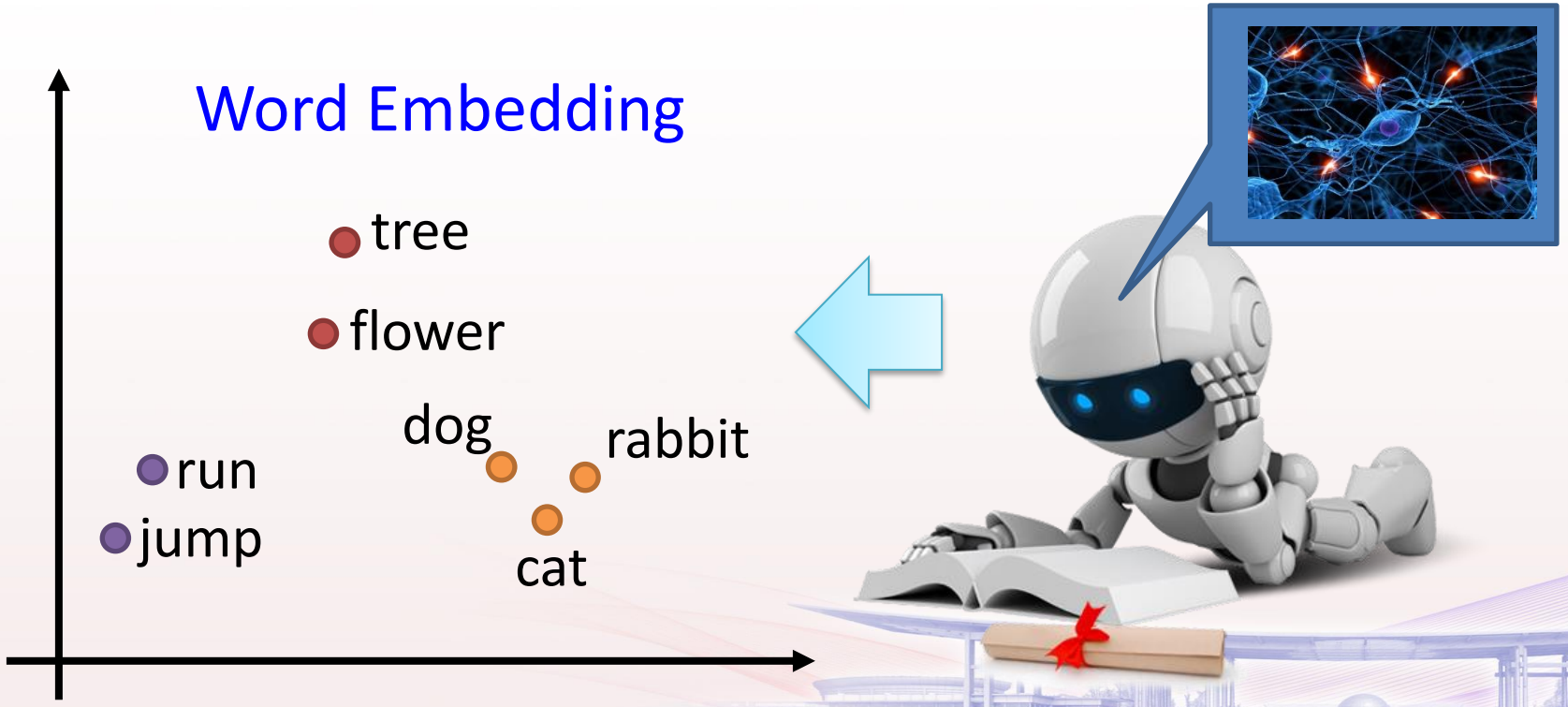
Word Embedding



Word Class

Word2Vec编码

- Machine learn the meaning of words from reading a lot of documents without supervision



Word2Vec编码

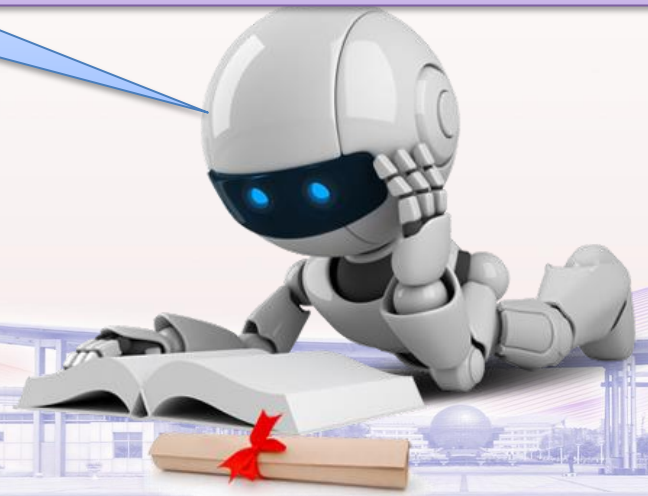
- Machine learn the meaning of words from reading a lot of documents without supervision
- A word can be understood by its context

蔡英文、馬英九 are something very similar

You shall know a word by the company it keeps

馬英九 520宣誓就職

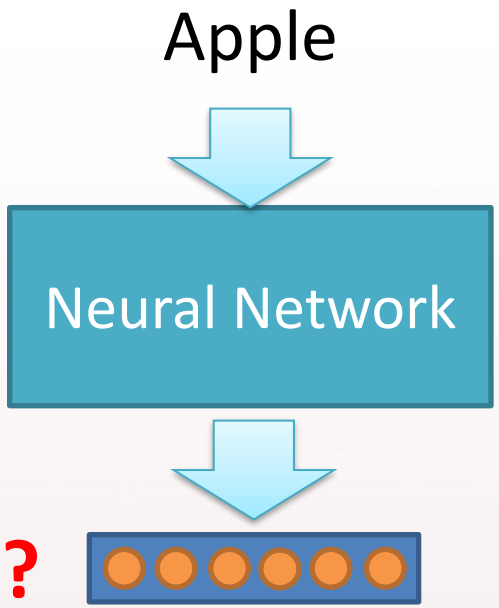
蔡英文 520宣誓就職



Word2Vec编码

How about auto-encoder?

- Generating Word Vector is unsupervised



Training data is a lot of text

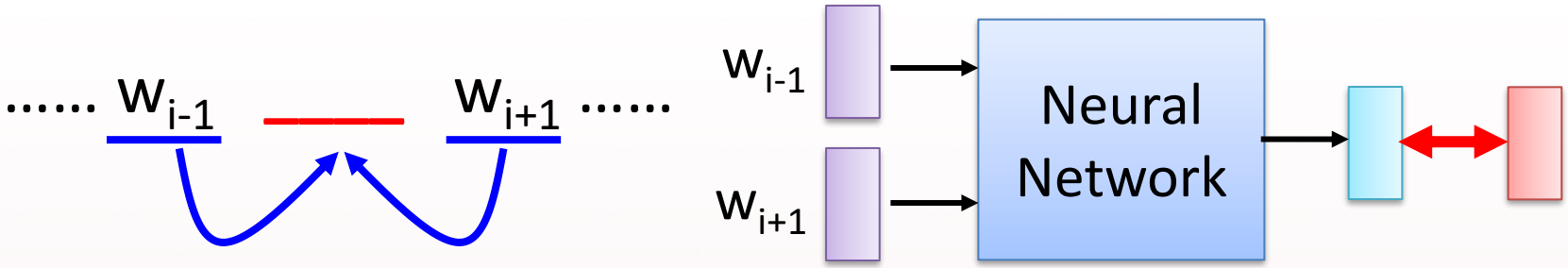
Word2Vec编码

Word2vec 的 2 种训练模式

1. CBOW(Continuous Bag-of-Words Model):

CBOW 的目标是根据上下文来预测当前词语的概率。

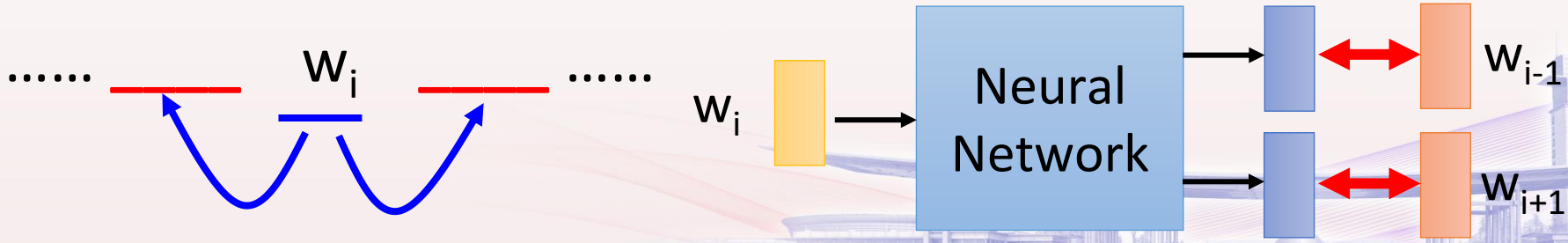
predicting the word given its context



2. Skip-gram (Continuous Skip-gram Model):

Skip-Gram恰好相反， 它是根据当前词语来预测上下文的概率。

predicting the context given a word



这两种方法都利用人工神经网络作为它们的分类算法。

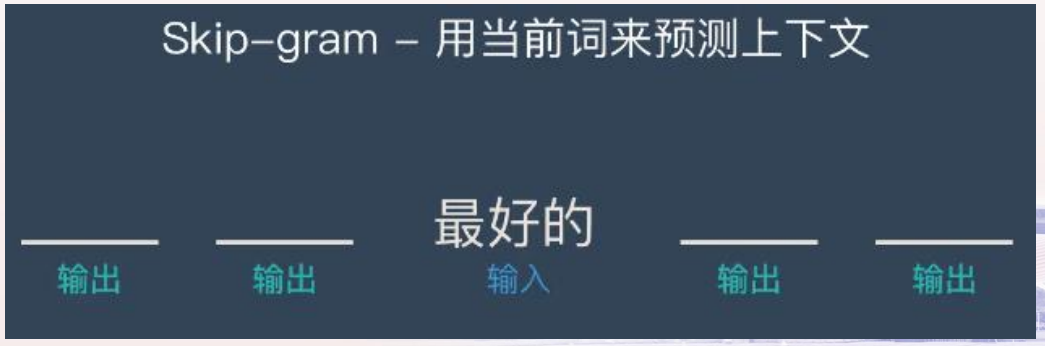
Word2Vec编码

■CBOW案例

通过上下文来预测当前值。相当于一句话中扣掉一个词，让你猜这个词是什么。

■ Skip-gram

用当前词来预测上下文。相当于给你一个词，让你猜前面和后面可能出现什么词。



Word2Vec编码

■Word2Vec优缺点：

优点：

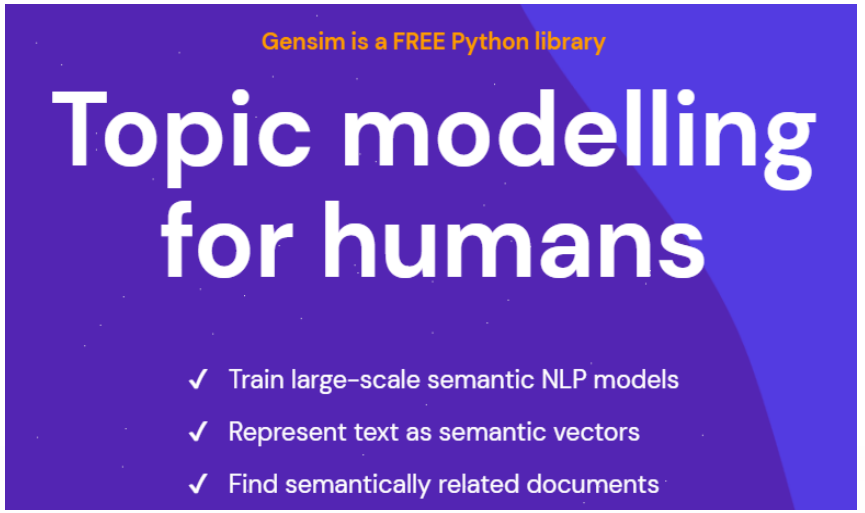
1. 由于 **Word2vec** 会考虑上下文，跟之前的 **Embedding** 方法相比，效果要更好（但不如 **18** 年之后的方法）
2. 比之前的 **Embedding**方 法维度更少，所以速度更快
3. 通用性很强，可以用在各种 **NLP** 任务中

缺点：

1. 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。
2. **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

https://code.google.com/archive/p/word2vec/

Word2Vec编码



■ Word2Vec实践：

1. Gensim

Gensim（generate similarity）是一个简单高效的自然语言处理Python库，用于抽取文档的语义主题（semantic topics）。Gensim的输入是原始的、无结构的数字文本（纯文本），内置的算法包括Word2Vec，FastText，潜在语义分析（Latent Semantic Analysis，LSA），潜在狄利克雷分布（Latent Dirichlet Allocation，LDA）等，通过计算训练语料中的统计共现模式自动发现文档的语义结构。这些算法都是非监督的，这意味着不需要人工输入——仅仅需要一组纯文本语料。一旦发现这些统计模式后，任何纯文本（句子、短语、单词）就能采用语义表示简洁地表达。

2.Jieba分词

jieba 主要用于Python中文分词，主要有以下3种特性：

a) 支持3种分词模式：精确模式、全模式、搜索引擎模式

b) 支持繁体分词

c) 支持自定义词典

https://radimrehurek.com/gensim/models/word2vec.html

Word2Vec编码

```
# -*- coding: utf-8 -*-
"""
项目名称：  word2vec模型训练
"""
import pandas as pd
import numpy as np
import jieba
import re
import codecs
import os
from gensim.models import word2vec
#首先将所有文件放在桌面的文件夹0000下
#==========注意以下两行需要改成你所要的文件路径和名称==
dir=os.path.dirname(os.path.abspath(__file__))
news=pd.read_csv("test.csv",encoding='gbk')
```

Word2Vec编码

```
class ClearData:
  def __init__(self, content):
    self.content = content
    pass
  #========提取中文字符=============
  def Extract(self,line):
      dr=re.compile(u'[\u4e00-\u9fa5]')
      return ''.join(dr.findall(line))
  def ExtractHanWen(self):
      ExtractData=[]
      for line in self.content:
          ExtractData.append(self.Extract(line))
      return ExtractData
  def __str__(self):
      print ('%s'%self.ExtractHanWen())

#==============分词=========================
class CutData(object):
    def __init__(self,text):
        self.cuttext=text
    pass
    def cutText(self):
        Text=[]
        for singleline in self.cuttext:
            wordSeg = jieba.cut(singleline, cut_all=False)
            Text.append(" ".join(wordSeg))
        return Text
    def __str__(self):
        print("%s"%self.cutText())
```

```
#==========去停用词==================
class DelStopWords(object):
    def __init__(self,data,stopwords):
        self.data=data
        self.stopwords=stopwords
        self.all_text=[]
        self.join_text=[]
    pass
    def Del(self):
        for line in self.data:
            single_text = []
            for word in line.split(' '):
                if '"' + word + '"' not in stopwords:
                    if len(word) >= 2:
                        single_text.append(word)
                    pass
                self.join_text = " ".join(single_text)
            self.all_text.append(self.join_text)
        return self.all_text
    def __str__(self):
        print("%s"%self.all_text)

def LoadStopWords():
    stopwords=[]
    filename='stopwords.txt'
    if os.path.exists(filename):
        st = codecs.open("stopwords.txt", "rb", encoding="gbk")
        for line in st:
            line = line.strip()
            stopwords.append(line)
    else:
        print("error:停词表不存在，请添加...")
    pass
    return stopwords
```

Word2Vec编码

```
class TrainWord2Vec():
    def train():
        print("开始清洗数据:")
        Data=ClearData(news['ART_CONTENT'])
        Cleared=Data.ExtractHanWen()
        cutData=CutData(Cleared).cutText()
        #print ('分词结果: ',cutData[0])
        print("...加载停用词表...")
        stopwords=LoadStopWords()
        print("...去停用词...")
        CorpusData=DelStopWords(cutData,stopwords).Del()
        sens_list=[]
        for data in CorpusData:
            #print('data:',data)
            sen=data.split(' ')
            sens_list.append(sen)
        print("...训练模型...")
        model=word2vec.Word2Vec(sens_list,iter=20,size=100)
        print('...存放模型...')
        model.save('./word2vec.model')

if __name__=='__main__':
    #print ('训练模型')
    #TrainWord2Vec.train()
    print('下载模型')
    model=word2vec.Word2Vec.load('word2vec.model')
    word_vectors={}
    for word in (model.wv.vocab):
        word_vectors[word]=model[word]
    print("美国和特朗普相似度",model.similarity("美国","特朗普"))
    print("中国和特朗普相似度",model.similarity("中国","特朗普"))
    print("基金和股票普相似度",model.similarity("基金","股票"))
    print("特朗普和股票普相似度",model.similarity("特朗普","股票"))
```

```
下载模型
美国和特朗普相似度 0.4763544
中国和特朗普相似度 0.17196551
基金和股票普相似度 0.2557576
特朗普和股票普相似度 0.039486375
```

谢谢观赏 下节课见