







Python 文本分析

• 目录

- 一  Python文本分析工具NLTK
- 二  分词
- 三  情感分析
- 四  文本分类
- 五  Word2vec
- 六  snownlp



一 Python文本分析工具NLTK

NLTK

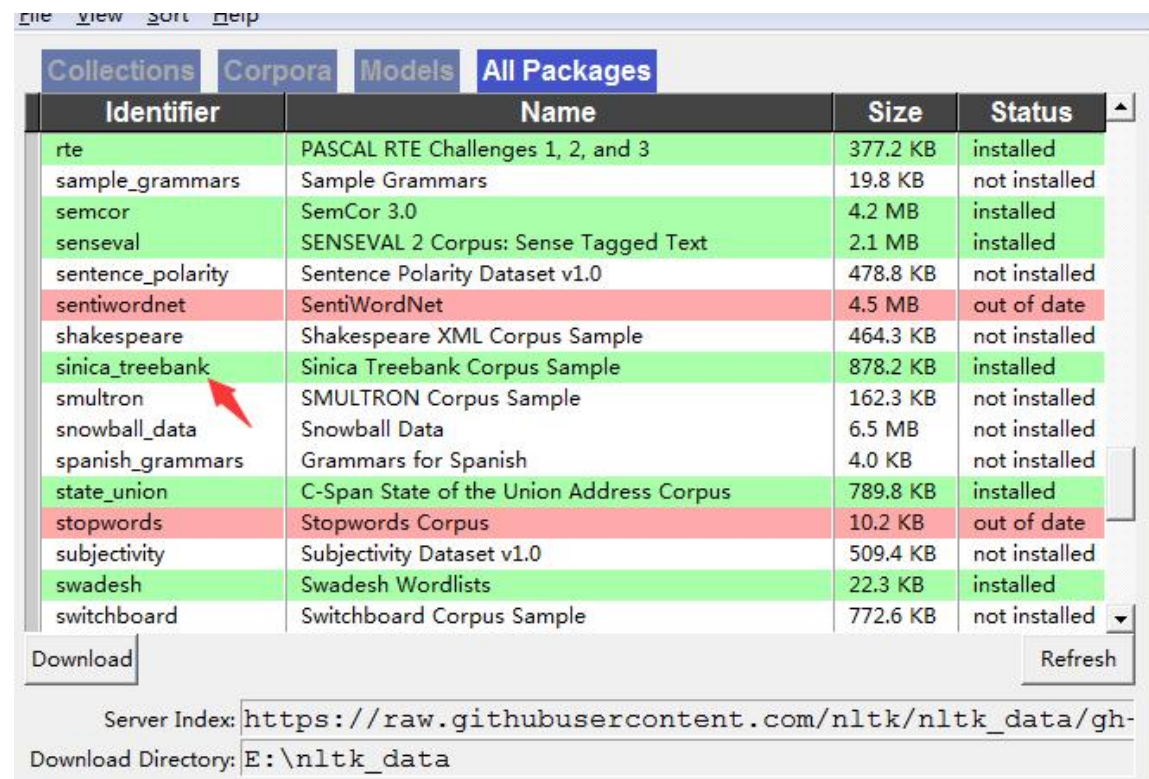
- Natural Language Toolkit，自然语言处理工具包，在NLP领域中，最常使用的一个Python库。
- NLTK是一个开源的项目，包含：Python模块，数据集和教程，用于NLP的研究和开发。
- NLTK由Steven Bird和Edward Loper在宾夕法尼亚大学计算机和信息科学系开发。

NLTK 安装

- Install NLTK:
 - `pip install -U nltk`
- Install Numpy (optional):
 - `pip install -U numpy`
- Test installation:
 - `python`
 - `import nltk`

NLTK 使用

- Python NLTK库中包含着大量的语料库，但是大部分都是英文，不过有一个Sinica（中央研究院）提供的繁体中文语料库，值得我们注意。
- `>>> import nltk`
- `>>> nltk.download()`



Identifier	Name	Size	Status
rte	PASCAL RTE Challenges 1, 2, and 3	377.2 KB	installed
sample_grammars	Sample Grammars	19.8 KB	not installed
semcor	SemCor 3.0	4.2 MB	installed
senseval	SENSEVAL 2 Corpus: Sense Tagged Text	2.1 MB	installed
sentence_polarity	Sentence Polarity Dataset v1.0	478.8 KB	not installed
sentiwordnet	SentiWordNet	4.5 MB	out of date
shakespeare	Shakespeare XML Corpus Sample	464.3 KB	not installed
sinica_treebank	Sinica Treebank Corpus Sample	878.2 KB	installed
smultron	SMULTRON Corpus Sample	162.3 KB	not installed
snowball_data	Snowball Data	6.5 MB	not installed
spanish_grammars	Grammars for Spanish	4.0 KB	not installed
state_union	C-Span State of the Union Address Corpus	789.8 KB	installed
stopwords	Stopwords Corpus	10.2 KB	out of date
subjectivity	Subjectivity Dataset v1.0	509.4 KB	not installed
swadesh	Swadesh Wordlists	22.3 KB	installed
switchboard	Switchboard Corpus Sample	772.6 KB	not installed

Download Refresh

Server Index: https://raw.githubusercontent.com/nltk/nltk_data/gh-

Download Directory: E:\nltk_data

NLTK数据文件结构

nlk 数据文件结构

```
[html]
1. nltk_data
2. |─ chunkers
3. |   └─ maxent_ne_chunker.zip
4. |─ corpora
5. |   └─ abc.zip
6. |   └─ alpino.zip
7. |   └─ biocreative_ppi.zip
8. |   └─ brown_tei.zip
9. |   └─ brown.zip
10. |   └─ cess_cat.zip
11. |   └─ cess_esp.zip
12. |   └─ chat80.zip
13. |   └─ city_database.zip
14. |   └─ cmudict.zip
15. |   └─ comtrans.zip
16. |   └─ conll12000.zip
17. |   └─ conll12002.zip
18. |   └─ conll12007.zip
19. |   └─ dependency_treebank.zip
20. |   └─ europarl_raw.zip
21. |   └─ floresta.zip
22. |   └─ gazetteers.zip
23. |   └─ genesis.zip
24. |   └─ gutenbergs.zip
25. |   └─ ieer.zip
26. |   └─ inaugural.zip
```

```
7. |   └─ wordnet_ic.zip
8. |   └─ wordnet.zip
9. |   └─ words.zip
10. |   └─ ycoe.zip
11. |─ grammars
12. |   └─ basque_grammars.zip
13. |   └─ book_grammars.zip
14. |   └─ large_grammars.zip
15. |   └─ sample_grammars.zip
16. |   └─ spanish_grammars.zip
17. |─ help
18. |   └─ tagsets.zip
19. |─ stemmers
20. |   └─ rslp.zip
21. |─ taggers
22. |   └─ averaged_perceptron_tagger.zip
23. |   └─ hmm_treebank_pos_tagger.zip
24. |   └─ maxent_treebank_pos_tagger.zip
25. |─ tokenizers
26. |   └─ punkt.zip
```

主要功能

语言处理任务	NLTK 模块	功能描述
获取和处理语料库	<code>nltk.corpus</code>	语料库和词典的标准化接口
字符串处理	<code>nltk.tokenize</code> , <code>nltk.stem</code>	分词, 句子分解提取主干
搭配发现	<code>nltk.collocations</code>	t-检验, 卡方, 点互信息 PMI
词性标识符	<code>nltk.tag</code>	n-gram, backoff, Brill, HMM, TnT
分类	<code>nltk.classify</code> , <code>nltk.cluster</code>	决策树, 最大熵, 贝叶斯, EM, k-means
分块	<code>nltk.chunk</code>	正则表达式, n-gram, 命名实体
解析	<code>nltk.parse</code>	图表, 基于特征, 一致性, 概率, 依赖
语义解释	<code>nltk.sem</code> , <code>nltk.inference</code>	λ 演算, 一阶逻辑, 模型检验
指标评测	<code>nltk.metrics</code>	精度, 召回率, 协议系数
概率与估计	<code>nltk.probability</code>	频率分布, 平滑概率分布
应用	<code>nltk.app</code> , <code>nltk.chat</code>	图形化的关键词排序, 分析器, WordNet 查看器, 聊天机器人
语言学领域的工作	<code>nltk.toolbox</code>	处理 SIL 工具箱格式的数据

使用NLTK比较文档相似度

案例实践



1

使用NLTK和jieba分

注意：碰到的是编码问题，最简单的解决办法是换成Python 3.X，没有中文的编码问题，中文默认是Unicode。如果是Python 2.7，要概对中文输入先解码(decode)成Unicode编码就好。



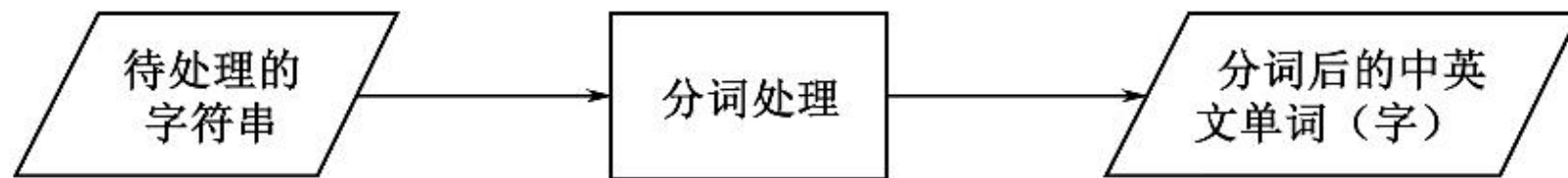
二 分词

分词 定义

文本分词是将字符串文本划分为有意义的单位的过程，如词语、句子或主题。

中文分词也叫作切分，是将中文文本分割成若干个独立、有意义的基本单位的过程。

分词算法基本的工作原理是根据输入的字符串文本进行分词处理、过滤处理，输出分词后的结果，包括英文单词、中文单词及数字串等一系列切分好的字符串。



分词原理图

为什么分词算法那么困难？

- 交集型歧义

- 武汉市长江大桥
- 乒乓球拍卖完了
- 已经取得文凭的和尚未取得文凭的干部
- 一次性生活补助

- 武汉市长/江大桥
- 乒乓球/拍卖完了
- 已经取得文凭的和尚/未取得文凭的干部
- 一次/性生活/补助

为什么分词算法那么困难？

- 组合型歧义

- 这个人 --- 个人恩怨
- 把手抬起来 --- 门把手坏了

- --小明，请用“难过”一词造句。
- --我家门前有一条小河很难过。
- --给我滚。

为什么分词算法那么困难？

- 未登录词
 - 然并卵
 - 令计划
 - 巴塞罗/那一场大雪
 - 佟大/为妻子生下一女儿

现有的中文分词算法可以分为以下3类：

1

基于词典的分词方法

它是将待处理的中文字符串与一个“尽可能全面”的词典中的词条按照一定的规则进行匹配，若某字符串存在于词典中，则认为该字符串匹配成功。

2

基于统计的分词方法

由于词是特定的字组合方式，那么在上下文中，相邻的单字共同出现的频率越高，则在该种字组合方式下就越有可能是构成了一个词。

3

基于理解的分词方法

该方法通过语义信息和语句信息来解决歧义分词问题，并且在分词的同时进行语义和句法分析。

各种分词方法的优劣对比表

分词方法	基于词典	基于理解	基于统计
歧义识别	差	强	强
新词识别	差	强	强
词库	需要	不需要	不需要
语料库	不需要	不需要	需要
规则库	不需要	需要	不需要
算法复杂性	容易	很难	一般
技术成熟度	成熟	不成熟	成熟
实施难度	容易	很难	一般
分词准确度	一般	准确	较准
分词速度	快	慢	一般

基于词典的算法

- 词典：哪些是词
- 最大匹配：每次都匹配最长的词
- Trie树

基于词典的算法

- 最少词数
- 独立自主/和平/等/互利/的/原则
- 独立自主/和/平等互利/的/原则
- 词典中添加“不成单字”的表。
- 他/说/的确/**实**/在理
- 他/说/的确/实在/**理**
- 他/说/的/确实/在理

基于词典的算法

- 最少词数+“不成单字表”
- 动态规划
- $\text{score}[i]$: 匹配到 i 结尾的最少罚分
- 对组合型歧义束手无策

基于理解的算法

- 同时完成分词与句法、语义分析等。
- Noam Chomsky 《句法结构》
- 句子 → 名词性短语 + 动词性短语
 - 名词性短语 → 名词
 - 名词性短语 → 形容词性短语 + 的 + 名词性短语
 - ...

基于统计的算法

- 最少词数+“不成单字表”
- 单词出现的频率
- 含大量真实语料的语料库

基于统计的算法

- 选择概率最大的分词方法

$$P(AB) = P(A|B)P(B)$$

- 句子：

$$S = W_1 W_2 W_3 \dots W_n$$

- 概率：

$$P(W_1 W_2 \dots W_n)$$

$$= P(W_1) * P(W_2|W_1) * \dots * P(W_n|W_1 W_2 \dots W_{n-1})$$

基于统计的算法

$$P(W_1) * P(W_2|W_1) * \dots * P(W_n|W_1 W_2 \dots W_{n-1})$$



$$P(W_1) * P(W_2) * \dots * P(W_n)$$



$$P(W_1) * P(W_2|W_1) * \dots * P(W_n|W_{n-1})$$

基于统计的算法

$$P(W_1) * P(W_2|W_1) * \dots * P(W_n|W_{n-1})$$

- 分词方法使概率最大？
- 动态规划

基于统计的算法

- 字标注法
- 把分词过程看做每个字在句子中的标注问题
- 例：4-tag(B-词首, M-词中, E-词尾, S-单独成词)
- 上/ B 海/ E 计/ B 划/ E 到/ S 本/ S 世/ B 纪/ E 末/ S
实/ B 现/ E 人/ B 均/ E 国/ B 内/ E 生/ B 产/ E 总/ B
值/ E 五/ B 千/ M 美/ M 元/ E。 / S

基于统计的算法

- 字标注法
- 各类机器学习算法
 - e.g. 条件随机场(CRF)
 - e.g. 支持向量机

MMSEG分词工具

MMSEG分词算法中包含了4种符合汉语语言中基本的成词习惯的歧义消解规则。



MMSEG分词算法中有两个重要的概念：**Chunk**和规则（**Rule**）。其中，一个**Chunk**就是一段字符串文本的一种分割方式，包括根据上下文分出的一组词及各个词对应的4个属性。规则的目的是过滤掉不符合特定要求的**Chunk**。为便于理解，我们可以将规则看做过滤器。

jieba分词工具

- 支持三种分词模式：
 - 1 精确模式，试图将句子最精确地切开，适合文本分析；
 - 2 全模式，把句子中所有的可以成词的词语都扫描出来，速度非常快，但是不能解决歧义；
 - 3 搜索引擎模式，在精确模式的基础上，对长词再次切分，提高召回率，适合用于搜索引擎分词。
- 支持繁体分词
- 支持自定义词典

jieba分词工具

- 分词方法说明

1 jieba.cut 方法接受三个输入参数: 需要分词的字符串; cut_all 参数用来控制是否采用全模式; HMM 参数用来控制是否使用 HMM 模型

2 jieba.cut_for_search 方法接受两个参数: 需要分词的字符串; 是否使用 HMM 模型。该方法适合用于搜索引擎构建倒排索引的分词, 粒度比较细

注意: 待分词的字符串可以是 unicode 或 UTF-8 字符串、GBK 字符串。注意: 不建议直接输入 GBK 字符串, 可能无法预料地错误解码成 UTF-8

jieba.cut 以及 jieba.cut_for_search 返回的结构都是一个可迭代的 generator, 可以使用 for 循环来获得分词后得到的每一个词语 (unicode), 或者用

3 jieba.lcut 以及 jieba.lcut for search 直接返回 list，建议使用list(cul())来转换，源码也是这样做的，少一步函数调用。

4 jieba.Tokenizer(dictionary=DEFAULT_DICT) 新建自定义分词器，可用于同时使用不同词典。jieba.dt 为默认分词器，所有全局分词相关函数都是该分词器的映射。

jieba分词使用

- >>>import jieba
- >>> print('/'.join(jieba.cut('如果放到post中将出错。',
HMM=False)))
- 如果/放到/post/中将/出错/



三 情感分析

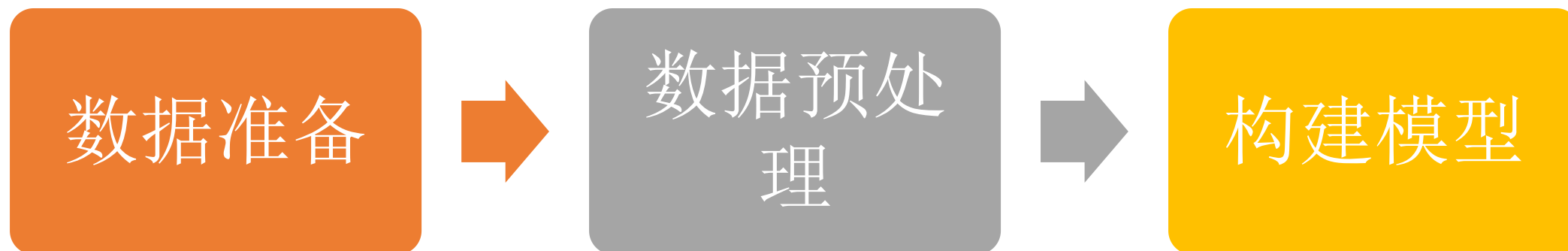
Python 情感分析之情感极性分析

- 「情感极性分析」是对带有感情色彩的主观性文本进行分析、处理、归纳和推理的过程。按照处理文本的类别不同，可分为基于新闻评论的情感分析和基于产品评论的情感分析。其中，前者多用于舆情监控和信息预测，后者可帮助用户了解某一产品在大众心目中的口碑。

情感极性两种分析方法

1. 基于情感词典的文本情感极性分析

- 通过情感打分的方式进行文本情感极性判断， $\text{score} > 0$ 判断为正向， $\text{score} < 0$ 判断为负向。



数据准备

- 情感词典及对应分数
 - BosonNLP数据下载的情感词典
 - 否定词词典
 - 程度副词词典 《知网》情感分析用词语集（beta版）
 - 停用词词典
- 词典把所有常用词都打上了唯一分数有许多不足之处。
 - 不带情感色彩的停用词会影响文本情感打分。
 - 由于中文的博大精深，词性的多变成为了影响模型准确度的重要原因。

有车一族都用了这个宝贝，后果很 严重 哦[偷笑][偷笑][偷笑]1，交警工资估计会打5折，没有超速罚款了[呲牙][呲牙][呲牙]2，移动联通公司大幅度裁员，电话费少了[呲牙][呲牙][呲牙]3，中石化中石油裁员2成，路痴不再迷路，省油[悠闲][悠闲][悠闲]5，保险公司裁员2成，保费折上折2成，全国通用[憨笑][憨笑][憨笑]买不买你自己看着办吧[调皮][调皮][调皮]

里面**严重等词**都是表达的相反意思，甚至整句话一起表示相反意思，不知死活的笔者还没能深入研究如何用词典的方法解决这类问题，但也许可以用机器学习的方法让神经网络进行学习能够初步解决这一问题。

- 同一个词可作多种词性，那么情感分数也不应相同，例如：

这部电影真 垃圾

垃圾 分类

- 很明显在第一句中垃圾表现强烈的贬义，而在第二句中表示中性，单一评分对于这类问题的分类难免有失偏颇。

预处理

- 分词

- Python常用的分词工具：

1. 结巴分词 Jieba

2. Pymmsseg-cpp

3. Loso

4. smallseg

- 去除停用词

- e.g. 这样/的/酒店/配/这样/的/价格/还算/不错

- --> 酒店/配/价格/还算/不错

简化的情感分数计算模型

- 所有情感词语组的分数之和

$$\text{finalSentiScore} = (-1)^{(\text{num of notWords})} * \text{degreeNum} * \text{sentiScore}$$

- 定义一个情感词语组：两情感词之间的所有否定词和程度副词与这两情感词中的后一情感词构成一个情感词组，即notWords + degreeWords + sentiWords，例如不是很交好，其中不是为否定词，很为程度副词，交好为情感词，那么这个情感词语组的分数为：
- $\text{finalSentiScore} = (-1)^1 * 1.25 * 0.747127733968$
- 其中1指的是一个否定词，1.25是程度副词的数值，0.747127733968为交好的情感分数。

问题：这个模型的缺点与局限性也非常明显：

- 首先，段落的得分是其所有句子得分的平均值，这一方法并不符合实际情况。正如文章中先后段落有重要性大小之分，一个段落中前后句子也同样有重要性的差异。
- 其次，有一类文本使用贬义词来表示正向意义，这类情况常出现与宣传文本中，还是那个例子：

有车一族都用了这个宝贝，后果很严重哦[偷笑][偷笑][偷笑]1，交警工资估计会打5折，没有超速罚款了[呲牙][呲牙][呲牙]2，移动联通公司大幅度裁员，电话费少了[呲牙][呲牙][呲牙]3，中石化中石油裁员2成，路痴不再迷路，省油[悠闲][悠闲][悠闲]5，保险公司裁员2成，保费折上折2成，全国通用[憨笑][憨笑][憨笑]
买不买你自己看着办吧[调皮][调皮][调皮]2980元轩辕魔镜带回家，推广还有返利[得意]

- 对于正负向文本的判断，该算法忽略了很多其他的否定词、程度副词和情感词搭配的情况；用于判断情感强弱也过于简单。

实践案例

- 2 基于情感词典的情感极性分析



2

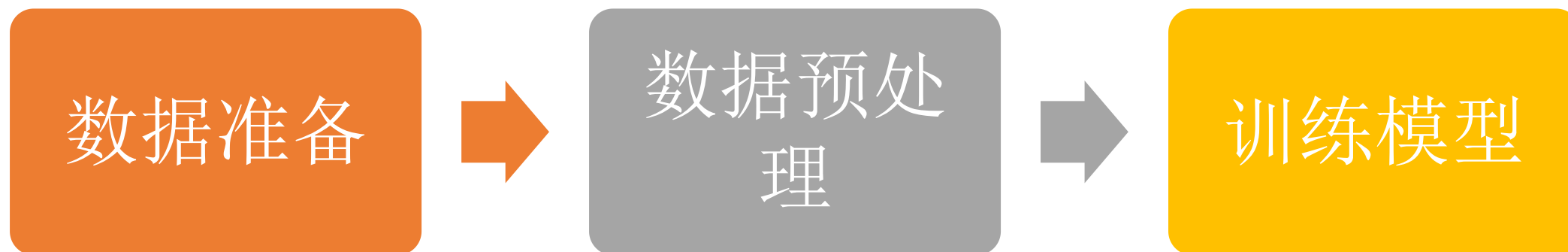
基于情感词典的情感



四 文本分类 五 Word2Vec

2 基于机器学习的文本情感极性分析

- 模型的输入需是数据元组，那么就需要将每条数据的词语组合转化为一个数值向量



Bag of Words (BOW)

One-hot Representation

向量中每个分量表示词典中对应单词在文档中出现的次数

1. Bob likes to play basketball, Jim likes too.
2. Bob also likes to play football games.

构造词典

Dict = {1: Bob, 2: like, 3: to, 4: play, 5: basketball, 6: also, 7: football, 8: games, 9: Jim, 10: too}

Vec1 = [1, 2, 1, 1, 1, 0, 0, 0, 1, 1]

Vec2 = [1, 1, 1, 1, 0, 1, 1, 1, 0, 0]

BOW

缺陷

“词汇鸿沟”现象：稀疏方式存储，其独立性假设不太符合语言文字实际分布情况，忽略了单词间的语法和顺序，无法了解单词间的关联程度

解决

- 采用SCPCD抽取整个短语
- 采用高阶（2阶以上）统计语言模型，例如bigram、trigram等

TF-IDF

如果某个词在一篇文章中出现的频率高，且在其他文章中出现频率低，则认为该词具有很好的类别区分能力，赋予更高权重

TF (Term Frequency)：给定词语在该文档中出现的次数

IDF (Inverse Document Frequency)：词语普遍重要性的度量

缺陷

单纯以词频作为单词重要性的度量，对于位置没有敏感性

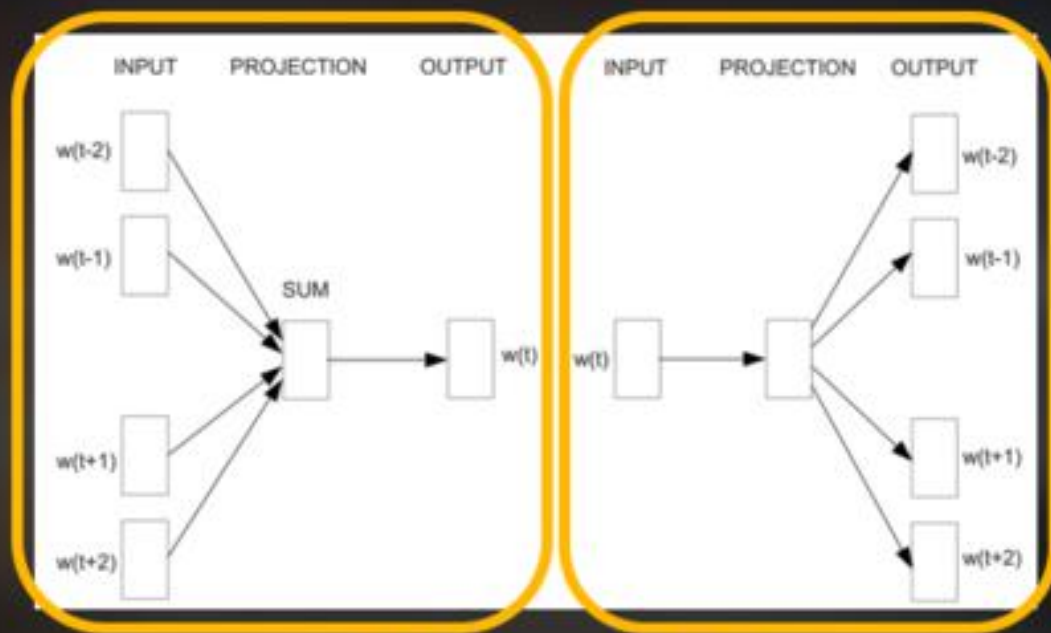
解决

人为添加权重

例如：第一段和最后一段赋予更高权重；每段第一句赋予更高权重

Word2Vec

词向量空间上的相似度可以用来表示文本语义上的相似度，常被用于词语聚类、找同义词、词性分析等



CBOW
(Continuous Bag-of-Words Model)

Skip-Gram
(with)

Word2Vec

CBOW : 根据上下文单词预测位置t的单词

$$P(w_t | w_{t-k}, w_{t-(k-1)} \dots, w_{t-1}, w_{t+1}, w_{t+2} \dots, w_{t+k})$$

Skip-Gram : 用于预测位置t的单词的上下文单词

$$P(w_i | w_t), \text{ 其中 } t - c \leq i \leq t + c \text{ 且 } i \neq t$$

特点

- 可进行向量的加法组合运算 (Additive Compositionality)

例如 : $\text{vec}('king') - \text{vec}('man') + \text{vec}('woman') = \text{vec}('queen')$

- 训练高效

模型的训练

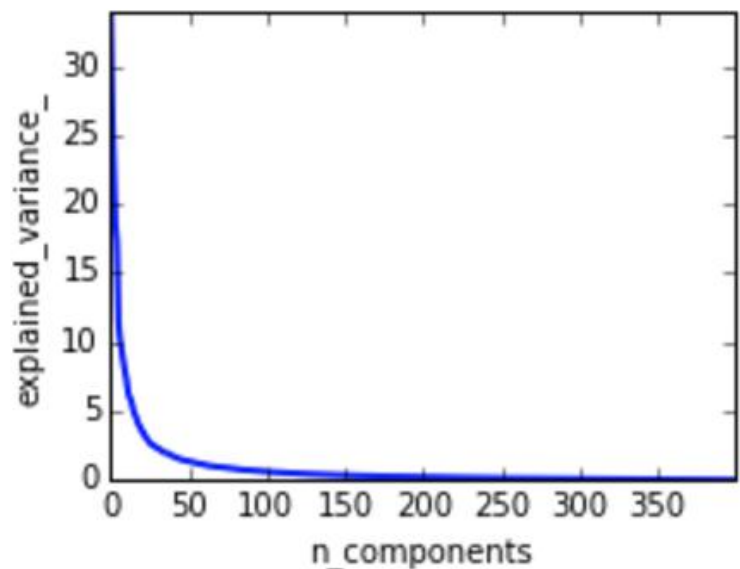


标准化

- 一般情况下： 标准化对模型的准确率影响不大
standardization
 $X = \text{scale}(X)$

PCA 降低维度

根据PCA结果，发现前100维能够cover 95%以上的variance



```
X_reduced = PCA(n_components = 100).fit_transform(X)
```

SVM (RBF) + PCA

- SVM (RBF)分类表现更为宽松，且使用PCA降维后的模型表现有明显提升，

```
clf = SVC(C = 2, probability = True)
```

```
clf.fit(X_reduced_train, y_reduced_train)
```

实践案例

- 3 基于机器学习的文本情感极性分析



3

基于机器学习的文本



六 snownlp 情感分析

SnowNLP

- SnowNLP是一个python写的类库，可以方便的处理中文文本内容，是受到了TextBlob的启发而写的，由于现在大部分的自然语言处理库基本都是针对英文的，
- SnowNLP方便处理中文的类库，并且和TextBlob不同的是，这里没有用NLTK，自带了一些训练好的字典。
- `pip install snownlp`
- <https://pypi.python.org/pypi/snownlp/0.11.1>

SnowNLP

- 1. 分词
- 2. 词性标注
- 3. 断句
- 4. 情绪判断
- 5. 拼音
- 6. 繁体转简体
- 7. 关键词抽取
- 8. 概括总结文意
- 9. 信息量衡量
- 10. 文本相似性

<http://www.jianshu.com/p/4692d1b5364d>

代码示例

```
from snownlp import SnowNLP
s = SnowNLP(u'这个东西真心很赞')
s.words # [u'这个', u'东西', u'真心',
# u'很', u'赞']
s.tags # [(u'这个', u'r'), (u'东西', u'n'),
# (u'真心', u'd'), (u'很', u'd'),
# (u'赞', u'Vg')]
s.sentiments # 0.9769663402895832 positive的概率
```

- 情感分析简单的将句子分为两类，积极和消极，即预测输入句子属于积极和消极的概率，sentiment属于[0,1]。越接近1表示正面情绪，越接近0表示负面情绪

训练自己的模型

```
from snownlp import seg
seg.train('data.txt')
seg.save('seg.marshal')#保存模型
from snownlp import tag
tag.train('199801.txt')
tag.save('tag.marshal')
from snownlp import sentiment
sentiment.train('neg.txt', 'pos.txt')
sentiment.save('sentiment.marshal')
```

NLP中文库

- jieba、snownlp、textgrocery
- 基本上分词都会用到jieba，速度快，分词挺准的