文本分类

CSDN学院
卿来云

自然语言处理

- 大量的信息以自然语言（英语、汉语等）形式存在
- 如何有效地获取和利用以自然语言形式出现的信息？
  - 自然语言处理（Natural Language Processing，NLP）：用计算机对语言信息进行处理的方法和技术
- NLP技术的应用
  - 机器翻译、自动摘要、文本分类、信息过滤、信息检索、信息抽取、文本挖掘、情感分析、自动问答、…

自然语言处理的主要任务

- 语言分析：分析语言表达的结构和含义
  - 词法分析：形态还原、词性标注、命名实体识别、分词（汉语、日语等）等
  - 句法分析：组块分析、结构分析、依存分析
  - 语义分析：词义、句义（逻辑、格关系、...）、篇章（上下文）（指代、实体关系）
- 语言生成：从某种内部表示生成语言表达
- 多语言处理（机器翻译、跨语言检索）：语言之间的对应、转换

自然语言处理的实现方法

- 基于规则的理性方法（Rationalist approach）
  - 基于以规则形式表达的语言知识（词、句法、语义以及转换、生成）进行推理
  - 强调人对语言知识的理性整理
  - 受Chomsky主张的人具有先天语言能力观点的影响，流行于1960－1985
- 基于语料库的经验方法（Empiricist approach）
  - 以大规模语料库（单语和双语）为语言知识基础
  - 利用统计学习和基于神经网络的深度学习方法自动获取和运用隐含在语料库中的知识
  - 学习到的知识体现为一系列模型参数

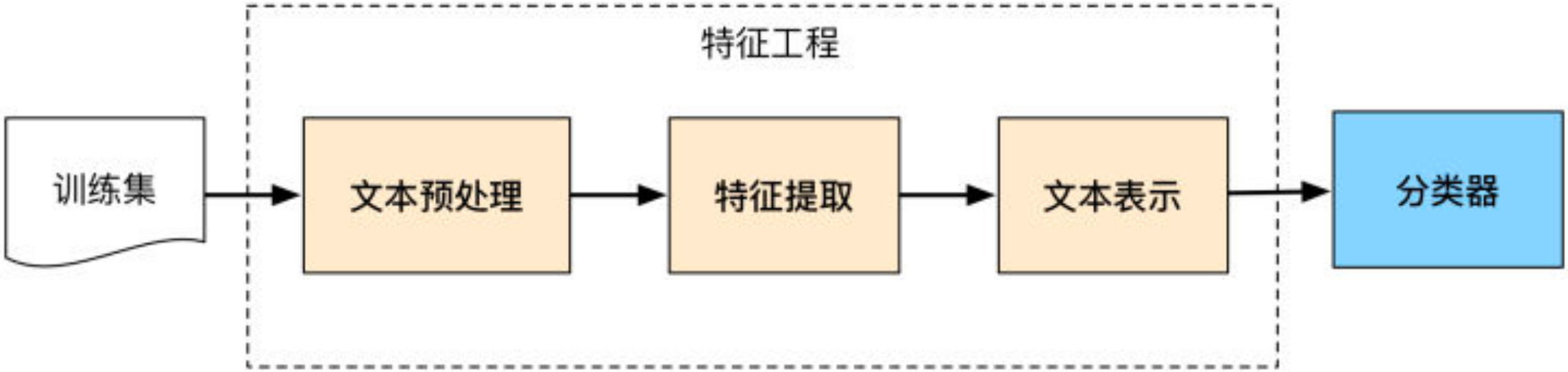
自然语言处理的难点

- 歧义处理
  - 有限的词汇和规则表达复杂、多样的对象
  - 苹果、粉丝、…
- 语言知识的表示、获取和运用
- 成语和惯用型的处理
- 对语言的灵活性和动态性的处理
  - 灵活性：同一个意图的不同表达，甚至包含错误的语法等
  - 动态性：语言在不断的变化，如：新词等
- 上下文和世界知识（常识，语言无关）的利用和处理

汉语处理的难点

- 缺乏计算语言学的句法/语义理论，大都借用基于西方语言的句法/语义理论
- 词法分析
  - 分词
  - 词性标注
- 句法分析
  - 主动词识别难
  - 词法分类与句法功能对应差
- 语义分析
  - 句法结构与句义对应差
  - 时体态确定难（汉语无形态变化）

文本分类



文本预处理

- 分词&词性标注

分词

- 传统算法：
  - 基于字符串匹配的正向/逆向/双向最大匹配
  - 基于理解的句法和语义分析消歧
  - 基于统计的互信息/CRF方法
- 深度学习：
  - WordEmbedding + Bi-LSTM+CRF方法逐渐成为主流

词法分析

- 形态还原（针对英语、德语、法语等）
  - 把句子中的词还原成基本词形
- 词性标注
  - 为句子中的词标上预定义类别集合（标注集）中的类
- 命名实体识别
  - 人名
  - 地名
  - 机构名
- 分词（针对汉语、日语等）
  - 识别出句子中的词

词性标注

- 为句子中的词标上预定义类别集合（标注集）中的类（词性），为后续的句法/语义分析提供必要的信息
- 开放类词性
  - Nouns、Verbs、Adjectives、Adverbs
- 封闭类词性
  - Determiners、Pronouns、Prepositions、Conjunctions、Auxiliary verbs、Particles（if、not、...）、Numerals

- 兼类词：一个词具有两个或者两个以上的词性
  - 把门*锁*上
  - 买了一把*锁*

词性标注方法

- 规则方法
  - 词典和规则提供候选词性
  - 消歧规则进行消歧
- 统计方法
  - 选择最可能的词性
  - 训练用语料库（已标注词性）
- 基于转换学习的方法
  - 统计学习得到规则
  - 用规则方法进行词性标注

汉语分词（切分）

- 词是语言中最小的能独立运用的单位，也是语言信息处理的基本单位。
- 分词是指根据某个分词规范，把一个“字”串划分成“词”串。
  - 难以确定何谓汉语的“词”
    - 单字词与语素的界定：猪肉、牛肉
    - 词与短语（词组）的界定：黑板、黑布
  - 信息处理用现代汉语分词规范：GB-13715（1992）
  - 具体应用系统可根据各自的需求制定规范
- 分词带来的问题
  - 丢失信息、错误的分词、不同的分词规范

切分歧义及歧义字段的种类

- 交集型歧义字段：ABC切分成AB/C或A/BC
  - 如：“和平等”
    - “独立/自主/和/平等/独立/的/原则”
    - “讨论/战争/与/和平/等/问题”
- 组合型歧义字段：AB切分成AB或A/B
  - 如：“马上”
    - “他/骑/在/马/上”
    - “马上/过来”
- 混合型歧义：由交集型歧义和组合型歧义嵌套与交叉而成
  - 如：“得到达”（交集型、组合型）
    - “我/今晚/得/到达/南京”
    - “我/得到/达克宁/了”
    - “我/得/到/达克宁/公司/去”

分词方法

一般通过分词词典和分词规则库进行分词。主要方法有：

- 正向最大匹配(FMM)或逆向最大匹配(RMM)
  - 从左至右(FMM)或从右至左(RMM)，取最长的词
  - “幼儿园 地 节目”或“幼儿 园地 节目”
- 双向最大匹配
  - 分别采用FMM和RMM进行分词
  - 如果结果一致，则认为成功；否则，
  - 采用消歧规则进行消歧（交集型歧义）：
- 正向最大、逆向最小匹配
  - 发现组合型歧义
- 逐词遍历匹配
  - 在全句中取最长的词，去掉之，对剩下字符串重复该过程
- 设立切分标记
  - 收集词首字和词尾字，把句子分成较小单位，再用某些方法切分
- 全切分
  - 获得所有可能的切分，选择最大可能的切分

词典

- 基础词典
  - 《现代汉语词典》第5版（2005年5月出版）：收录了65，000个词

- 特定应用：
  - 预处理：“去停止词”
  - 训练集中所有出现的词
  - 训练集中出现次数高于阈值的词
  - 训练集中出现次数最高的K个词

Jieba分词

- 项目主页：https://github.com/fxsjy/jieba

- 支持三种分词模式
  - 精确模式：试图将句子最精确地切开，适合文本分析（项目推荐使用）
  - 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
  - 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

调用

- import jieba

- trainFile = codecs.open(trainFileName, 'rb', encoding='utf-8')
- i = 0
- for line in trainFile:
  - i += 1
  - line = line.strip()
  - sep_index = line.index(',')
  - label = line[:sep_index]
  - line = line[sep_index + 1:]
  - words = jieba.cut(line)

jieba.cut("我来到北京清华大学", cut_all=False)

支持用户自定义词典

系统自带词典：dict.txt

- 开发者可以指定自己自定义的词典
  - jieba.load_userdict(file_name)
  - file_name 为文件类对象或自定义词典的路径
- 词典格式： dict.txt 一样
  - 一个词占一行
  - 每一行分三部分：词语、词频（可省略）、词性（可省略），用空格隔开，顺序不可颠倒。
- file_name 若为路径或二进制方式打开的文件，则文件必须为 UTF-8 编码。
- 词频省略时使用自动计算的能保证分出该词的词频

► 例：自定义词典

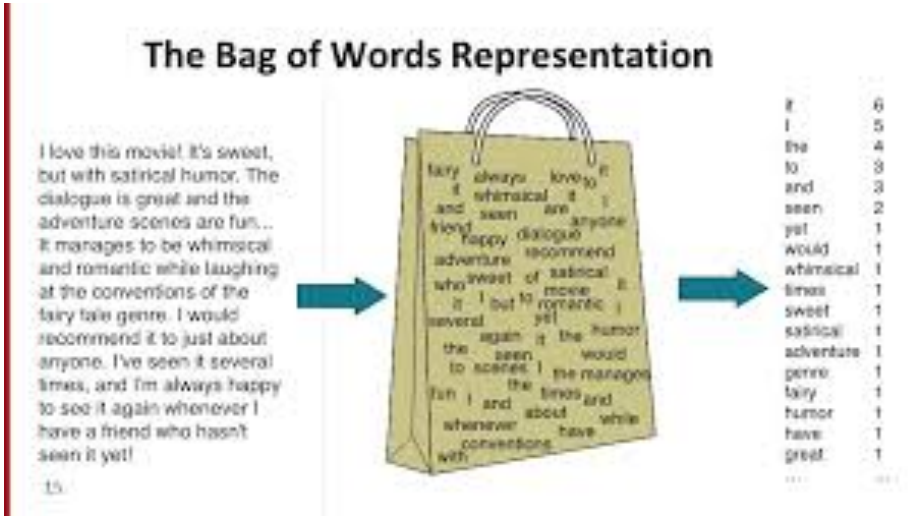
- 创新办 3 i
- 云计算 5
- 凯特琳 nz
- 台中

文本分类

- 文本分类：机器对文本按照一定的分类体系自动标注类别的过程，其定义如下：
  - *输入*: 文档 d
  - 文档类别： $C = \{c_1, c_2, \ldots, c_J\}$
  - *输出*：文档d对应的类别 c（*或文档*d为类别c的概率$p(c|d)$）
- 文本分类问题与其它分类问题没有本质上的区别
  - 特征表示
  - 分类器

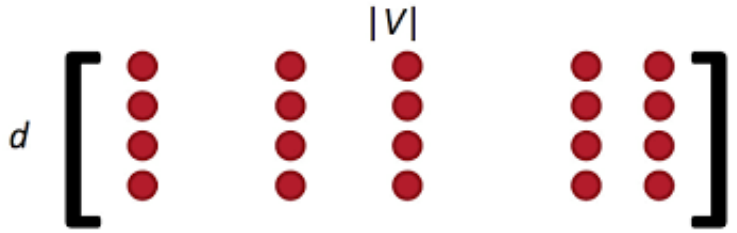
文本特征表示

- 文档：一系列单词
  - 1. 词袋模型：每个单词表示为一个数值
    - 二值：单词是否出现
    - 整数：词频
    - 连续值：TF-IDF

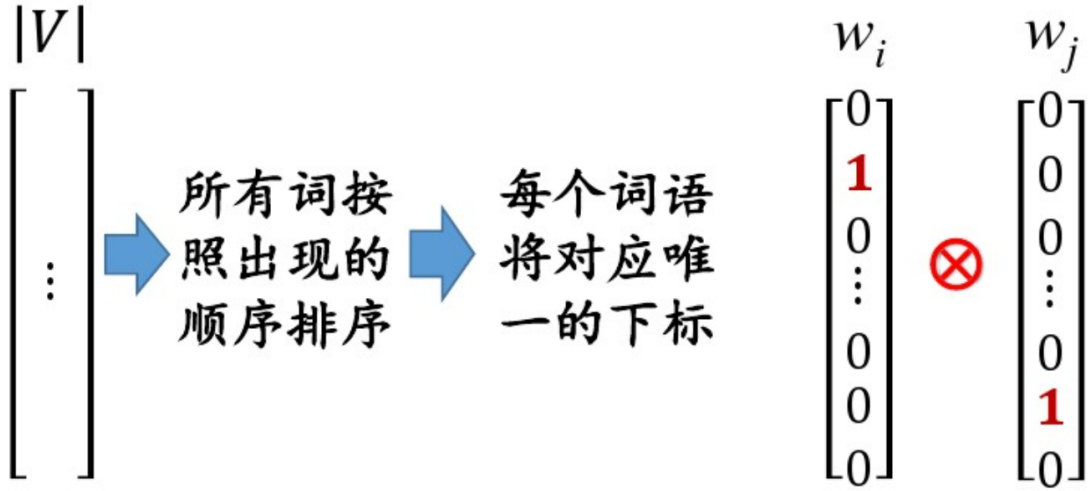


sklearn.feature_extraction.text.TfidfVectorizer

文本特征表示

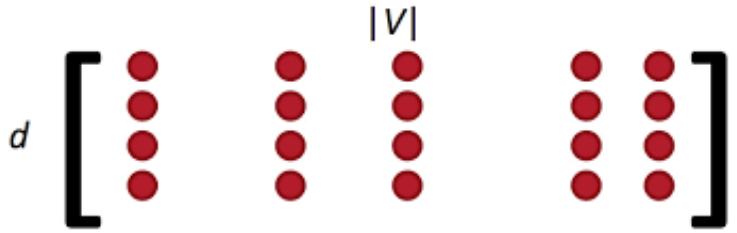


- 文档：一系列单词
  - 2. 词向量模型：每个单词表示为一个向量
  - One-hot：

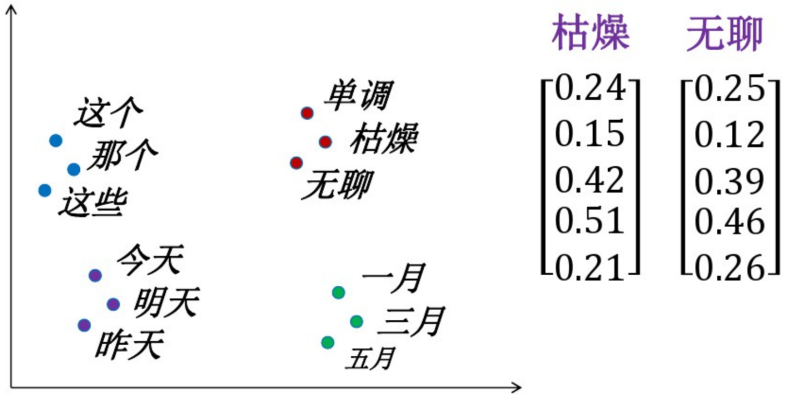


任意两个词之间的相似度都为0！

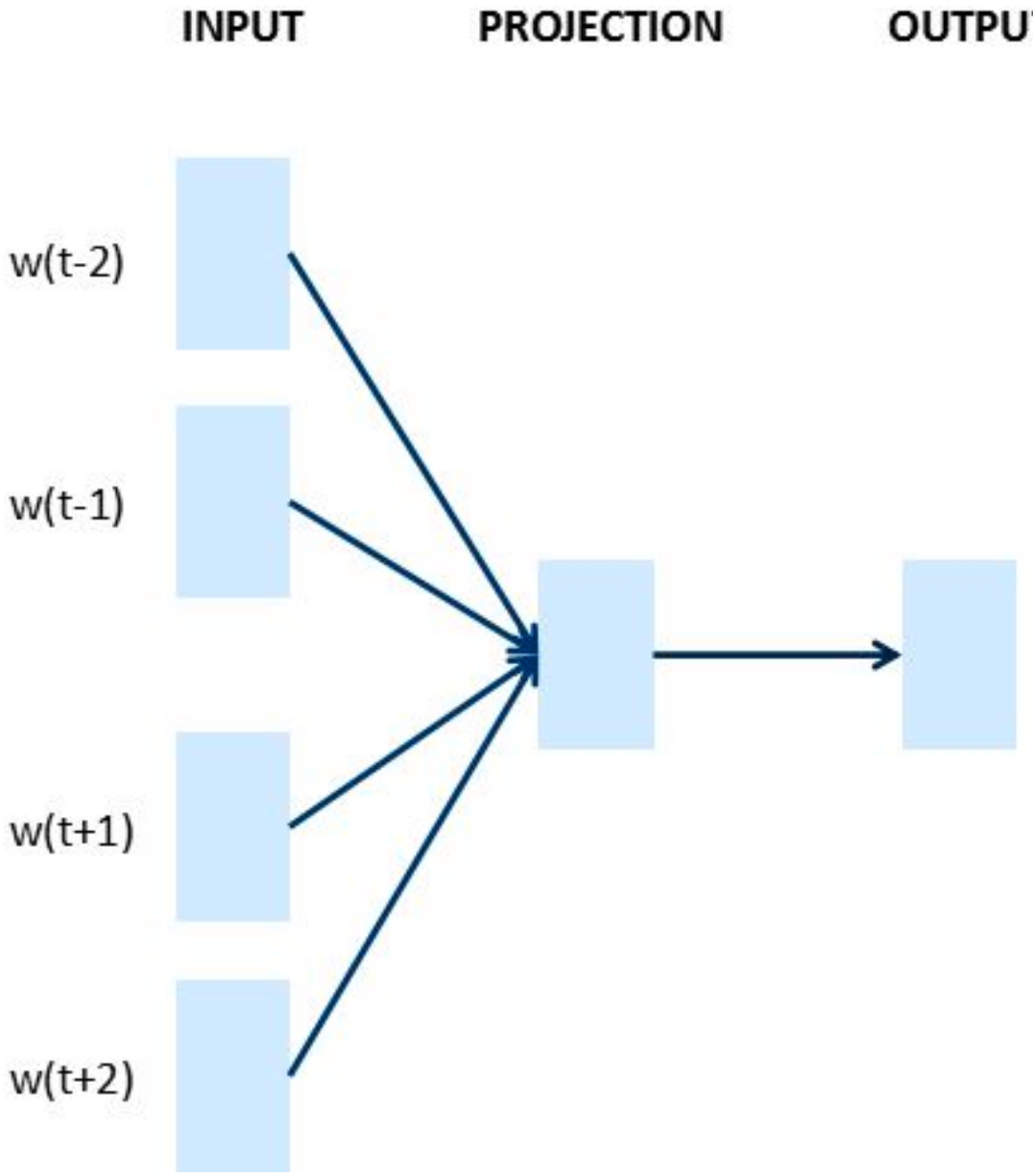
文本特征表示



- 文档：一系列单词
  - 2. 词向量模型：每个单词表示为一个向量
  - One-hot
  - word2vec：根据词的使用情况/上下文
    - 低维、稠密的实数空间



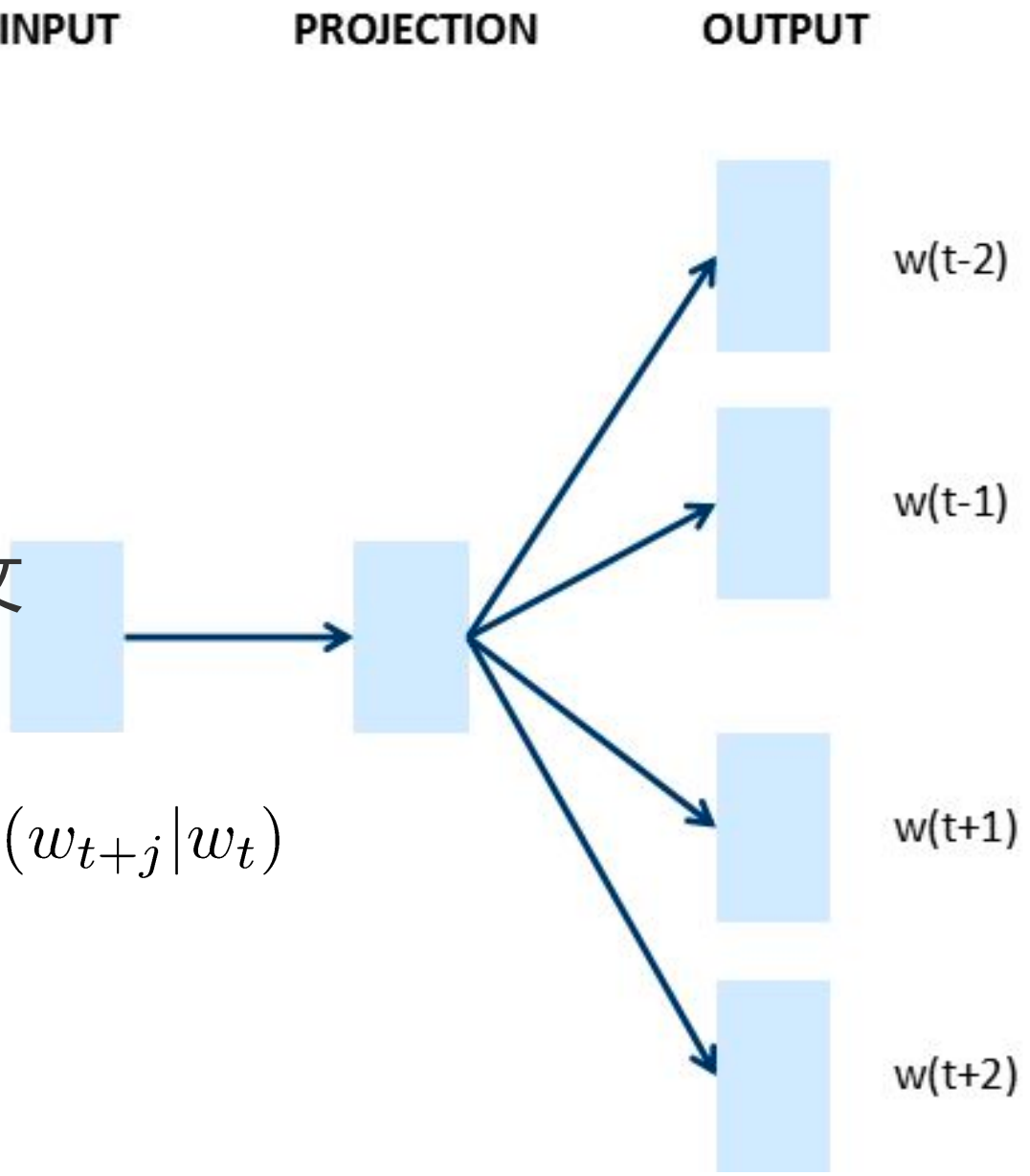
CBOW：上下文来预测当前词



CBOW对小型数据库比较合适

Skip-gram：当前词预测上下文

$$J(\theta) = \frac{1}{T}\sum_{t=1}^{T} \sum_{-m \le j \le m, j \neq 0} \log p(w_{t+j}|w_t)$$



Skip-Gram在大型语料中表现更好

- 目标函数：$J(\theta) = \frac{1}{T}\sum_{t=1}^{T}\sum_{-m\leq j\leq m, j\neq 0} \log p(w_{t+j}|w_t)$
- 其中 $p(o|c) = \frac{\exp\left(u_o^T v_c\right)}{\sum_{w=1}^{W}\exp\left(u_w^T v_c\right)}$
  - c：中心词，v为中心词词向量
  - o：外围词，u为外围词词向量
  - 每个词语有两个词向量
  - 动态Logistic回归

word2vec

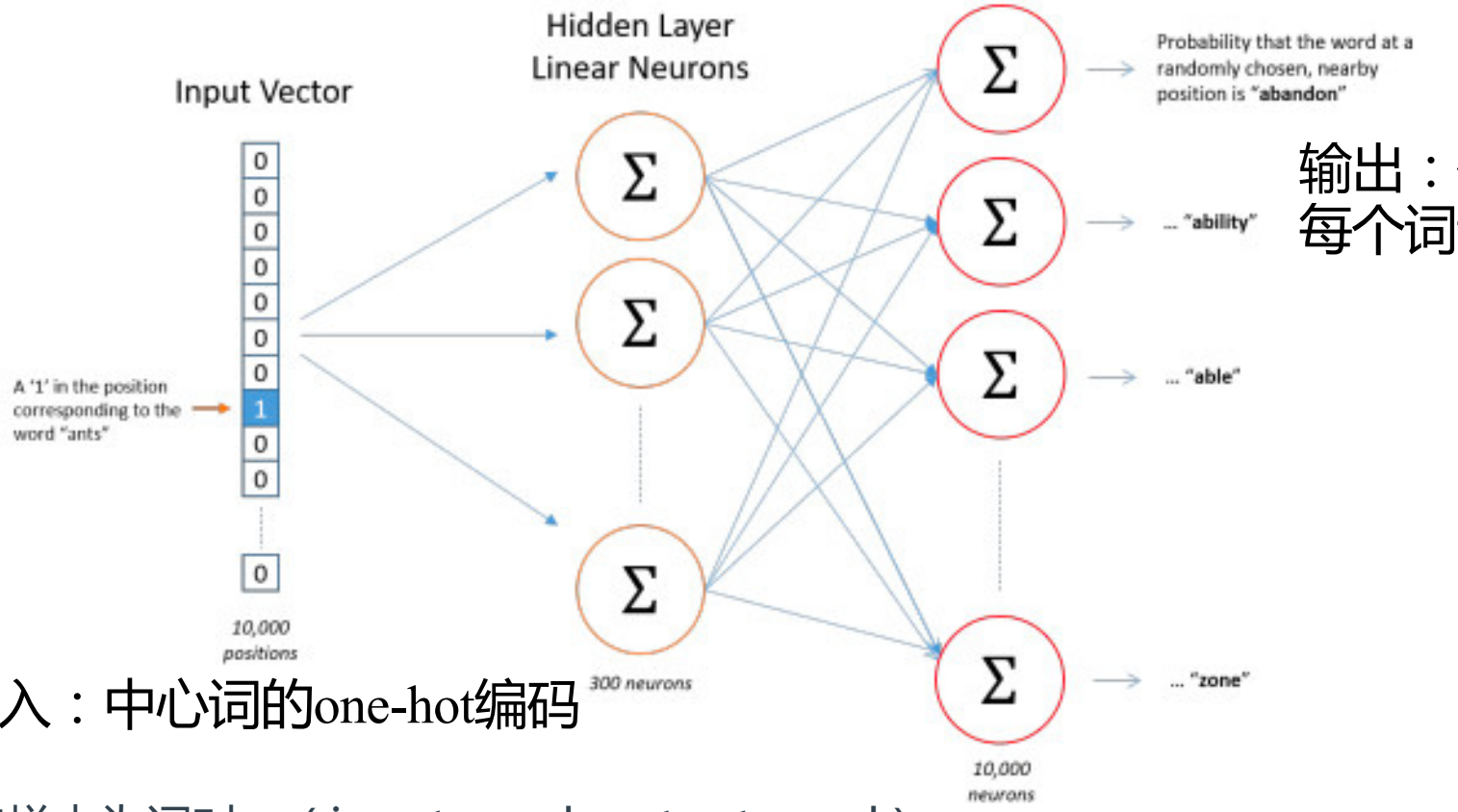
- word2vec 是 Google 于 2013 年开源推出的一个用于获取 word vector 的工具包: 简单、高效

https://github.com/zhyq/word2vec-google
https://code.google.com/archive/p/word2vec/

网络框架

隐层没有使用
任何激活函数

输出层使用了sotfmax



输出：外围词为
每个词语的概率

输入：中心词的one-hot编码

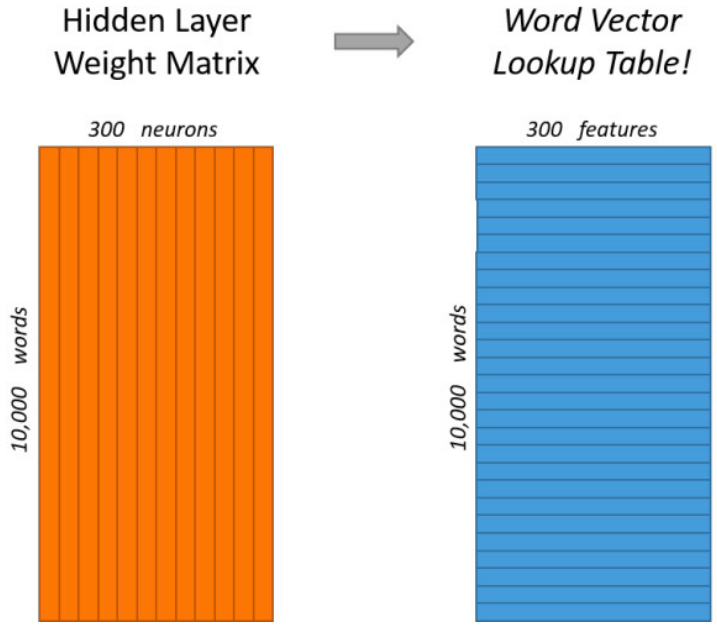
训练样本为词对：（input word, output word）

隐含层

- 隐含节点的数目为词向量的维数：100 ~ 500
- 隐含权重矩阵
  - 假设隐含节点数目为300
  - 输入向量维数（字典大小）为10000
  - 则隐含权重矩阵为10000行，300列

左边：每一列代表一个10000维的词向量和隐层单个神经元连接的权重向量

右边：每一行代表每个单词的词向量



隐含层

- 输入向量与隐含层权重矩阵相乘：

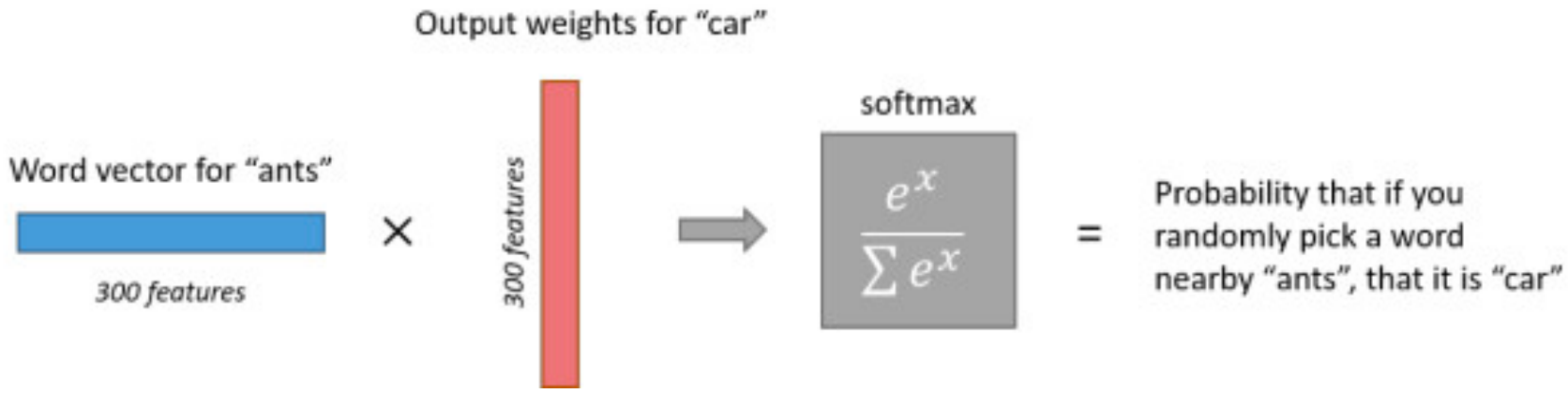
$$[0 \quad 0 \quad 0 \quad 1 \quad 0] \times \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ 10 & 12 & 19 \\ 11 & 18 & 25 \end{bmatrix} = [10 \quad 12 \quad 19]$$

选择矩阵中对应的向量中维度值为1的索引行

隐层权重矩阵便成了一个”查找表“（lookup table）：
进行矩阵计算时，直接去查输入向量中取值为1的维度下对应的那些权重值。隐层的输出就是每个输入单词的“嵌入词向量”。

输出层

- 例：训练样本为 (input word: “ants”， output word: “car”)



两个不同的单词有着非常相似的“上下文”，
那么这两个单词的嵌入向量将非常相似

训练

- 隐含层权重矩阵非常大
  - 如假设隐含节点数目为300，输入向量维数（字典大小）为10000，则隐含权重矩阵为10000行，300列（300万）
- 训练策略：
  - 将常见的单词组合（word pairs）或者词组作为单个“words”处理
  - 对高频次单词进行抽样，减少训练样本的个数
  - 对优化目标采用“negative sampling”方法，这样每个训练样本的训练只会更新一小部分的模型权重，从而降低计算负担
    - 一个训练样本（词对）的10000维输出中，只有一维接近1，其余均接近0，对接近0的输出权重只采样少量权重进行呢更新（如5个）

各种工具包

版本	地址	CBOW		Skip-Gram	
C	http://word2vec.googlecode.com/svn/trunk/	HS	NEG	HS	NEG
python	http://radimrehurek.com/gensim/			HS	
Java	https://github.com/ansjsun/Word2VEC_java	HS		HS	
C++	https://github.com/jdeng/word2vec	未知	未知	未知	未知

► 工具包：gensim

- 安装：pip install gensim
- 使用：
  - **from** gensim.models **import** Word2Vec
  - model = Word2Vec(sentences, sg=1, size=100, window=5, min_count =5, negative=3, sample=0.001, hs=1, workers=4)

官网：https://radimrehurek.com/gensim/models/word2vec.html
使用： http://blog.csdn.net/john_xyz/article/details/54706807

Word2Vec(sentences, sg=1, size=100, window=5, min_count=5, negative=3, sample=0.001, hs=1, workers=4)

参数	说明
sg：算法	1.sg=1是skip-gram算法，对低频词敏感；默认sg=0为CBOW算法
size：输出词向量的维数	值太小会导致词映射因为冲突而影响结果，值太大则会耗内存并使算法计算变慢，一般值取为100到200之间
window：当前词与目标词之间的最大距离	3表示在目标词前看3-b个词，后面看b个词（b在0-3之间随机）
min_count：过滤掉频率小于min-count的单词	
negative：负采样	
sample：高频率词被随机下采样到所设置的阈值	
hs：是否使用层级softmax	1表示使用 hs=0且negative不为0，则使用负采样
workers	训练的并行，此参数只有在安装了Cpython后才有效，否则只能使用单核

保存模型

- model.save(fname)
- model = Word2Vec.load(fname)

分类器

- 朴素贝叶斯算法（Naive Bayes）
- Logistic回归
- SVM算法
- 集成学习
  - xgboost/LightGBM
- 深度神经网络
  - 卷积神经网络(TextCNN)
  - 循环神经网络(TextRNN)
  - TextRNN+Attention
  - TextRCNN(TextRNN+CNN)

朴素贝叶斯分类器

- 朴素贝叶斯分类器：数据挖掘十大经典算法”之一
  - 假设在给定类别的情况下，特征之间独立：
  - $p(\mathbf{x}|y=c)=\prod_{j=1}^{D}p(x_j|y=c)$
  - 则根据贝叶斯公式：$$p(y=c|\mathbf{x})=\frac{p(\mathbf{x}|y=c)p(y=c)}{\sum_{c'}p(\mathbf{x}|y=c')p(y=c')}$$
  - 通常多个小概率相乘容易溢出→在log域中加法
- 文本分类中，文本通常被表示成词袋模型的形式：假设文本中每个词汇的出现相互独立，并且不考虑出现的先后顺序，则可用朴素贝叶斯分类器进行文档分类，其中x_j表示单词j的特征
- Scikit-learn中，根据单词的表示形式（二值/整数/连续值），可用BernoulliNB /MultinomialNB/ GaussianNB 。

http://scikit-learn.org/stable/modules/naive_bayes.html

朴素贝叶斯分类器

- 模型训练：根据训练集计算模型参数
- 例：采用BernoulliNB分类器，则
  - $p(x_j|y=c)=\frac{N_{cj}}{\sum_{j\prime\in V}N_{cj\prime}}$，其中$N_{cj}$表示单词$j$在类别$c$的文档出现的次数，$V$为词典
    - Smoothing：如果某词语在训练集中没有出现，会得到概率为0（乘以其他单词的概率也为0），因此增加平滑：$p(x_j|y=c)=\frac{N_{cj}+1}{\sum_{j\prime\in V}N_{cj\prime}+|V|}$
  - $p(x_j|y=c)=\frac{D_c}{D}=\frac{D_c}{\sum_{c\prime}D_{c\prime}}$，其中$D_c$表示类别$c$的文档出现的次数

建议时间安排

- 第一周：熟悉数据，完成分词
- 第二周：完成特征提取
- 第三周：baseline模型（分类器选择）
- 第四周&第五周：模型迭代

reference

- http://cs224d.stanford.edu/