《人工智能软件开发与实践》

（ 2023 学年 秋季 学期）

作

业

报

告

学　　号：_

姓　　名：__

班　　级：____

任课教师：___

作业报告

实验名称： **中文词怎样表示（1）**

成绩：

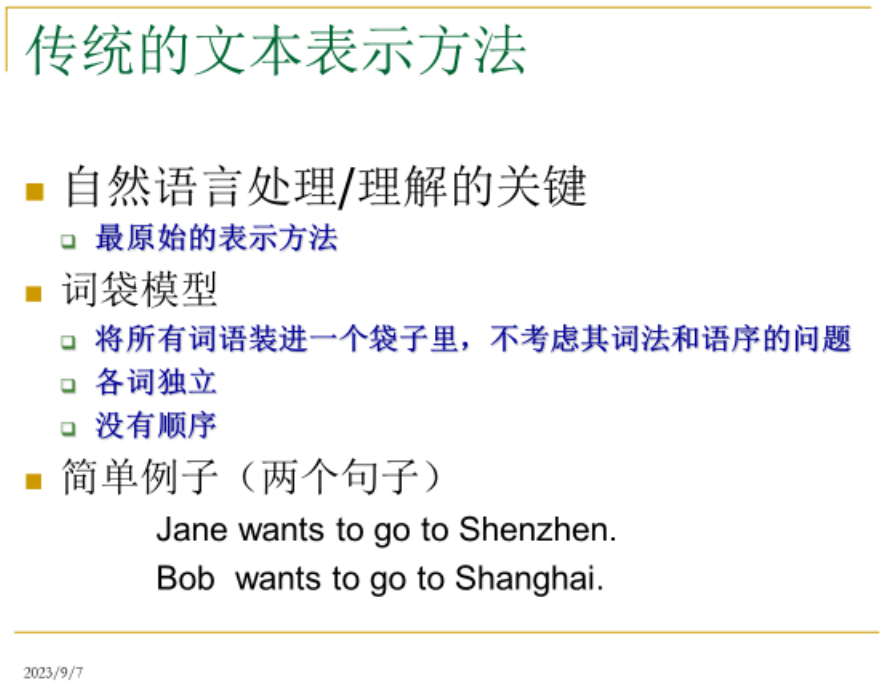
实验类别： 验证/综合型实验 **实验要求：** 1 人 1 组 **时间：** 2023 年 9 月 8 日

一、 实验目的

了解中文词的表示方式。了解词袋模型的缺点，词向量表示的优点。掌握使用 gensim 库进行词袋表示，并进行 word2vec 的训练。

二、实验内容

词袋模型和向量表示（NLP 的巨人肩膀 https://zhuanlan.zhihu.com/p/50443871?from=singlemessage ）



传统的文本表示例子

Unigram

- 词袋(8个词)
  - [Jane, wants, to, go, Shenzhen, Bob, Shanghai,.]
- 两个句子表示为8维：

[1,1,2,1,1,0,0,1]

[0,1,2,1,0,1,1,1]

  - 8维，第2行第5维的‘0’是什么意思？
- 简单例子（两个句子）

Jane wants to go to Shenzhen.

Bob wants to go to Shanghai.

2023-9-7

传统的文本表示例子

Bigram

- bigram词袋(理论上64个，实际上9个）
  - [Jane wants, wants to, to go, go to, to Shenzhen, Shenzhen ., Bob wants, to Shanghai, Shanghai .]
- 两个句子表示为（9维）：

[1,1,1,1,1,1,1,1,1]

[0,1,1,1,0,0,1,1,1]

  - 如果Uni-gram和Bi-gram

torch.cat([1,1,2,1,1,0,0,1],[1,1,1,1,1,1,1,1,1])

torch.cat([0,1,2,1,0,1,1,1],[0,1,1,1,0,0,1,1,1])

2023/9/7

把词的表示从向量的一个维度，变成一个向量（参考图像中的一个点，表示成 RGB 三原色）

如果图像中没有RGB

- 怎么表示图像
  - 用2^{16}表示RGB的每个可能
  - 每个点有$3*2^{16}$颜色可能
  - 一张图（$1024*768*3*2^{16}$）
- 缺点
  - 表示空间大了很多
  - 很难衡量两种颜色的相似程度
  - 向量的维度之间相互独立
  - 0-1特征或整数特征值，不连续、不可导
  - 无法直接应用大多数机器学习和最优化方法

寻找语义的“基”向量

- 假设有一个300维的语义基向量
  - 每一个词都很可以表示为一个300维的向量
  - 不用训练，直接就可以得到词的相似度
- 方法
  - LSI（隐性语义检索，奇异值分解SVD和主成分分析PCA）
  - PLSI（概率隐性语义检索）
  - LDA（Latent Dirichlet Allocation，DPMM）
  - 非参数贝叶斯
  - Deep
  - 张量

2003 年时 Bengio et al. 发表了 A neural probabilistic language model，首次提出了神经语言模型，这篇文章也被看作是词嵌入技术的开端。

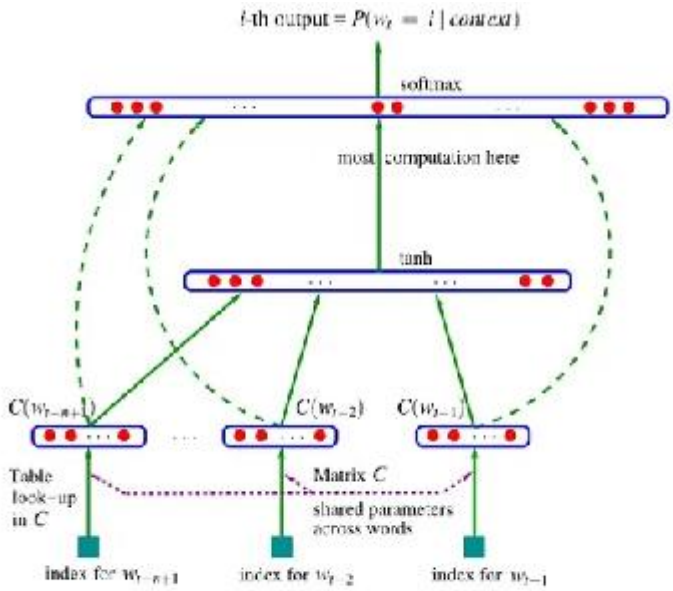


Figure 1: Neural architecture: $f(i, w_{t-1}, \cdots, w_{t-n+1}) = g(i, C(w_{t-1}), \cdots, C(w_{t-n+1}))$ where g is the neural network and $C(i)$ is the i-th word feature vector.

word2vec 是 Google 于 2013 年推出的， 分为 3 层：input layer, hidden layer(projection layer), output layer。模型训练就是为了学习两个矩阵：input word embedding(输入与隐层之间的权重矩阵） 和 output word embedding（隐层与输出之间的权重矩阵），这样每个词就对应两个词向量：输入词向量 v, 输出词向量 u。有两种具体方法:

(1) CBOW(Continuous Bag of Words）：由上下文预测中心词，为每个词学习 2 个向量：输入向量 v(作为上下文词时的词向量)，输出向量 u(作为中心词时的词向量)。

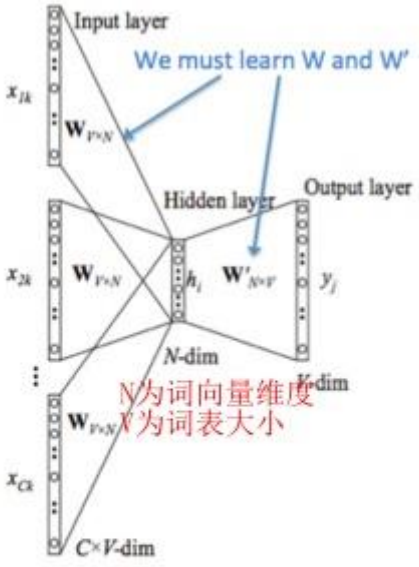


Figure 1: This image demonstrates how CBOW works and how we must learn the transfer matrices 词向量矩阵

(2) skip-gram: 由中心词预测上下文，为每个词学习 2 个向量：输入向量 v(作为中心词时的词向量)，输出向量 u(作为上下文词时的词向量)。

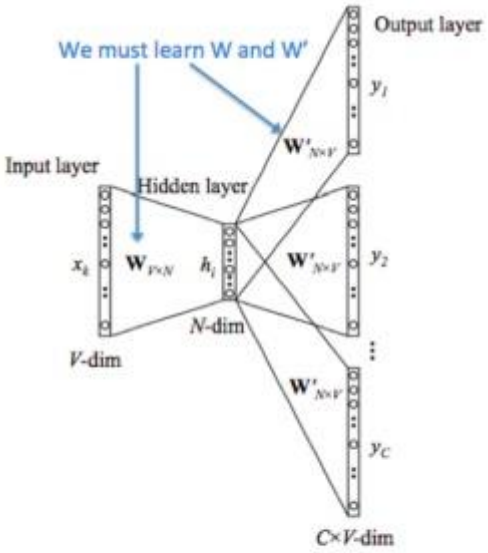


Figure 2: This image demonstrates how Skip-Gram works and how we must learn the transfer matrices

使用 Gensim 库完成词的词袋表示和 word2vec 的训练使用。

实验 1：gensim corpora dictionary 使用

(1) 将字符串切分为单词，并去掉停用词(stoplist = set('for a of the and to in'.split(' ')))

(2) 统计每个词的出现频度，只出现 1 次的词，作为噪音去掉

(3) 使用 gensim.corpora.Dictionary 库将字符串转化为 id 的列表

实验 2：使用 gensim.word2vec 训练模型并使用

(1) 装载数据

(2) 训练 word2vec 模型

(3) 保存模型

(4) 装载模型

(5) 使用 word2vec 模型计算两个词的相似度

(6) 计算某个词的相关词列表

(7) 寻找一个词的对应关系

(8) 寻找不合群的词

(9) 查看词向量

(10) 寻找比第二个词更接近第一个词的词

Word2Vec 参数说明

https://blog.csdn.net/HiWangWenBing/article/details/121723021

三、使用的算法名称（若无，可以不填）

四、程序源码（拷贝至此处，同时作为附件和报告再一份单独的程序）

实验 1：

```
# import pprint
text_corpus = [
    "Human machine interface for lab abc computer applications",
    "A survey of user opinion of computer system response time",
    "The EPS user interface management system",
    "System and human system engineering testing of EPS",
    "Relation of user perceived response time to error measurement",
    "The generation of random binary unordered trees",
    "The intersection graph of paths in trees",
    "Graph minors IV Widths of trees and well quasi ordering",
    "Graph minors A survey",
]
# Create a set of frequent words
stoplist = set('for a of the and to in'.split(' '))
# Lowercase each document, split it by white space and filter out stopwords
texts = [[word for word in document.lower().split() if word not in stoplist]
         for document in text_corpus]

# Count word frequencies
from collections import defaultdict
frequency = defaultdict(int)
for text in texts:
    for token in text:
        frequency[token] += 1

# Only keep words that appear more than once
processed_corpus = [[token for token in text if frequency[token] > 1] for text in texts]
# pprint.pprint(processed_corpus)

from gensim import corpora

dictionary = corpora.Dictionary(processed_corpus)
print(dictionary)
print(dictionary.token2id)
```

```
bow_corpus = [dictionary.doc2bow(text) for text in processed_corpus]
print(bow_corpus)
```

实验 2：

```
# -*- coding: utf-8 -*-
# @Time : 2019/11/13 14:55
# @FileName: word2vec-gensim.py
# @Software: PyCharm
# @Author : yip
# @Email : 522364642@qq.com
# @Blog : https://blog.csdn.net/qq_30189255
# @Github : https://github.com/yip522364642


import warnings

warnings.filterwarnings("ignore")

'''
1 获取文本语料并查看
'''
# with open('text8', 'r', encoding='utf-8') as file:
#     for line in file.readlines():
#         print(line)

'''
2 载入数据，训练并保存模型
'''
from gensim.models import word2vec
# from gensim import corpora
# import logging
# logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s', level=logging.INFO)  # 输出日志信息
sentences = word2vec.Text8Corpus('text8')  # 将语料保存在 sentence 中

#word2vec.corpora.dictionary(sentences)
# model = word2vec.Word2Vec(sentences, sg=1, vector_size=100,  window=5,  min_count=5,  negative=3, sample=0.001, hs=1, workers=4)  # 生成词向量空间模型
# model.save('text8_word2vec.model')  # 保存模型
# dictionary = corpora.Dictionary('text8'.split(' '))


'''
3 加载模型，实现各个功能
'''
# 加载模型
```

```
model = word2vec.Word2Vec.load('text8_word2vec.model')
#打印每个词对应的 ID
for index, word in enumerate(model.wv.index_to_key):
    print(str(index)+":"+word)
# 计算两个词的相似度/相关程度
print("计算两个词的相似度/相关程度")
word1 = u'man'
word2 = u'woman'
result1 = model.wv.similarity(word1, word2)

print(word1 + "和" + word2 + "的相似度为：", result1)
print("\n================================")

# 计算某个词的相关词列表
print("计算某个词的相关词列表")
word = u'bad'
result2 = model.wv.most_similar(word, topn=10)  # 10 个最相关的
print("和" + word + "最相关的词有：")
for item in result2:
    print(item[0], item[1])
print("\n================================")

# 寻找对应关系
print("寻找对应关系")
print(' "boy" is to "father" as "girl" is to ...? ')
result3 = model.wv.most_similar(['girl', 'father'], ['boy'], topn=3)
for item in result3:
    print(item[0], item[1])
print("\n")

more_examples = ["she her he", "small smaller bad", "going went being"]
for example in more_examples:
    a, b, x = example.split()
    predicted = model.wv.most_similar([x, b], [a])[0][0]
    print("'%s' is to '%s' as '%s' is to '%s'" % (a, b, x, predicted))
print("\n================================")

# 寻找不合群的词
print("寻找不合群的词")
result4 = model.wv.doesnt_match("flower grass pig tree".split())
print("不合群的词：", result4)
print("\n================================")

# 查看词向量（只在 model 中保留中的词）
print("查看词向量（只在 model 中保留中的词）")
word = 'girl'
```

```
print(word, model.wv[word])
# for word in model.wv.vocab.keys():  # 查看所有单词
#     print(word, model[word])

#寻找比第二个词更接近第一个词的词
word1 = 'first'
word2 = 'last'
result5 = model.wv.closer_than(word1, word2)
NOR = min(10, len(result5))
print("比%s 更接近%s 的%d 个词有：" % (word2, word1, NOR), end='')
for i in range(NOR):
    if not i == NOR - 1:
        print(result5[i] + ', ', end='')
    else:
        print(result5[i] + '。')


'''
4 增量训练
'''
model = word2vec.Word2Vec.load('text8_word2vec.model')
more_sentences = [['Advanced','users','can','load','a','model','and','continue','training',
'it', 'with', 'more', 'sentences']]
model.build_vocab(more_sentences, update=True)
model.train(more_sentences, total_examples=model.corpus_count, epochs=1)
model.save('text8_word2vec.model')
```

五、程序运行结果（将程序运行结果的截图拷贝至此处，或者填写实验结果）

实验 1：

Dictionary<12 unique tokens: ['computer', 'human', 'interface', 'response', 'survey']...>

{'computer': 0, 'human': 1, 'interface': 2, 'response': 3, 'survey': 4, 'system': 5, 'time': 6, 'user': 7, 'eps': 8, 'trees': 9, 'graph': 10, 'minors': 11}

[[(0, 1), (1, 1), (2, 1)], [(0, 1), (3, 1), (4, 1), (5, 1), (6, 1), (7, 1)], [(2, 1), (5, 1), (7, 1), (8, 1)], [(1, 1), (5, 2), (8, 1)], [(3, 1), (6, 1), (7, 1)], [(9, 1)], [(9, 1), (10, 1)], [(9, 1), (10, 1), (11, 1)], [(4, 1), (10, 1), (11, 1)]]

实验 2：（关闭了打印每个词和其对应 id）

计算两个词的相似度/相关程度

man 和 woman 的相似度为： 0.6812836

==========

计算某个词的相关词列表

和 bad 最相关的词有：

good 0.805596649646759

luck 0.7330778241157532

horrid 0.6796099543571472

foolish 0.6583592295646667

stupid 0.6574098467826843

ugly 0.6557650566101074

somebody 0.6446459293365479

nasty 0.6424608826637268

pretty 0.6420729160308838

happy 0.6380300521850586

==========

寻找对应关系

"boy" is to "father" as "girl" is to ...?

mother 0.7603307366371155

wife 0.7168700098991394

grandmother 0.6705162525177002

'she' is to 'her' as 'he' is to 'his'

'small' is to 'smaller' as 'bad' is to 'worse'

'going' is to 'went' as 'being' is to 'was'

==========

寻找不合群的词

不合群的词： pig

==========

查看词向量（只在 model 中保留中的词）

girl [-8.49663243e-02 6.35894597e-01 1.17972687e-01 -4.18373555e-01
 -4.95625287e-02 -9.60830301e-02 -2.98992217e-01 4.45590466e-01
 -2.03770116e-01 -1.82172477e-01 -3.27504575e-01 1.96300641e-01
 -6.06770277e-01 5.57170928e-01 8.68723541e-03 -5.23133814e-01
 -5.50896078e-02 -5.70445613e-04 9.58015174e-02 -2.60244280e-01
  8.00825730e-02 -2.14545541e-02 -4.51815277e-01 4.60632533e-01
 -1.86505049e-01 2.83226192e-01 5.70604987e-02 -3.88708711e-01
 -2.15654641e-01 2.84347665e-02 1.77874148e-01 -1.18496560e-01
  7.89833888e-02 -3.52008343e-01 -2.52918929e-01 -3.72817777e-02
 -1.99752226e-01 -7.30042085e-02 -2.67093509e-01 5.21020172e-03
 -3.11058432e-01 2.69434787e-02 -1.12294160e-01 1.29137605e-01
  2.14934517e-02 -5.99525981e-02 4.32102829e-02 4.00378406e-02
  5.10792434e-02 7.65974745e-02 1.73066229e-01 -2.84297522e-02
 -3.80609870e-01 4.09149826e-02 -1.02054939e-01 1.72397673e-01
  2.01499999e-01 -5.71979210e-02 -3.21930081e-01 -2.03907117e-01
  5.92110217e-01 -9.23579037e-02 1.09965868e-01 1.03676029e-01
 -2.44966313e-01 2.85479993e-01 1.38543203e-01 1.00737065e-01
 -2.81800061e-01 -4.04000908e-01 -7.55432770e-02 -1.41085625e-01
  2.55848262e-02 7.25158006e-02 3.51437449e-01 1.14888899e-01
  4.31038618e-01 -1.98075041e-01 -3.32059979e-01 1.61274239e-01
  2.64046013e-01 -3.05331230e-01 -1.83344066e-01 2.69162655e-01
 -1.06126361e-01 4.05205727e-01 4.28871870e-01 3.56979370e-01
  4.10863534e-02 2.36147001e-01 1.87474787e-01 -1.20127209e-01
  6.61605656e-01 -1.64924085e-01 9.64649394e-02 -1.34339944e-01
 -2.15680629e-01 4.09226686e-01 1.11737013e-01 -1.07628837e-01]

比 last 更接近 first 的 1 个词有：second。

六、心得体会和遇到的困难

了解了自然语言处理的发展过程，学习到了几个经典模型的实践使用。