文本相似度

汇报人：孙天艺　张新田　苏向迎　高振波　孙润庚

信息爆炸

信息过载

在信息爆炸时代，人们迫切希望从海量信息中获取与自身需要和兴趣吻合度高的内容。为了满足此需求，出现了多种应用，如搜索引擎、自动问答系统、文档分类与聚类、文献查重、文献精准推送等，而这些应用场景的关键技术之一就是文本相似度计算技术。

人类兴趣

引 言

CONTENTS

1 相关概念
Introduction

2 计算方法
Algorithm

3 算法示例
Demo

北京理工大學
BEIJING INSTITUTE OF TECHNOLOGY

相关概念

PART ONE

1 相关概念 | 文本相似度

相似度定理[1]：

$$Sim\ (A,B)\ = \frac{logP(common(A,B))}{logP(description(A,B))} \tag{1-1}$$

用于没有限制应用领域，此定义是较多被采用的概念

[1]Lin D. An Information-theoretic Definition of Similarity [C]// Proceedings of the 15th International Conference on Machine Learning.1998

1 相关概念 | 相关度

相关度：

两个事物以任何形式相互关联的程度

包括上下位关系、同义关系、反义关系、部件-整体关系、值-属性关系等

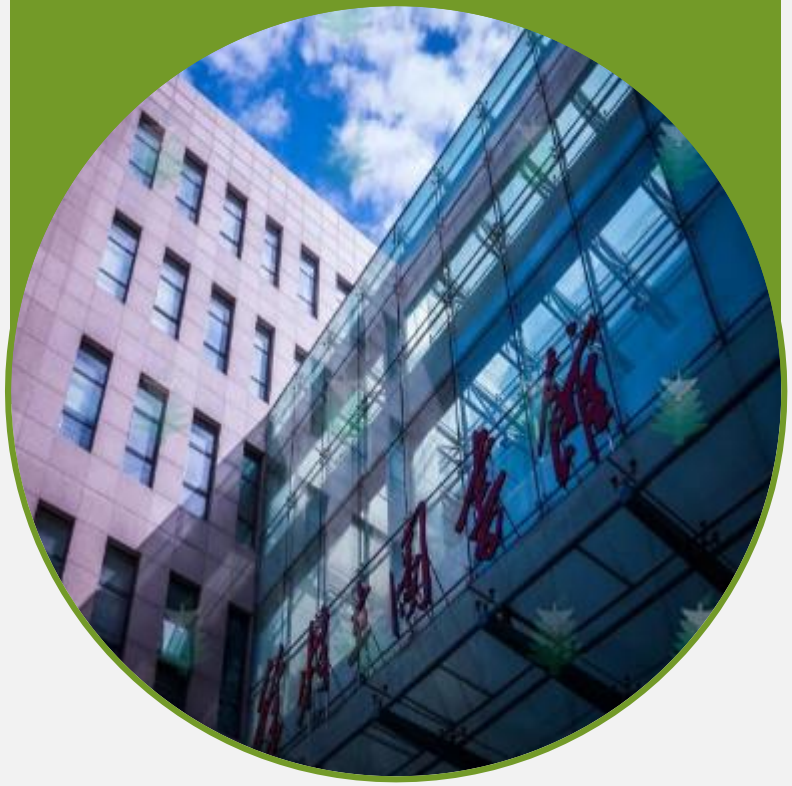
相似度是相关度的一种特殊情况，包括上下位关系、同义关系

文本相似度越高，相关度越大

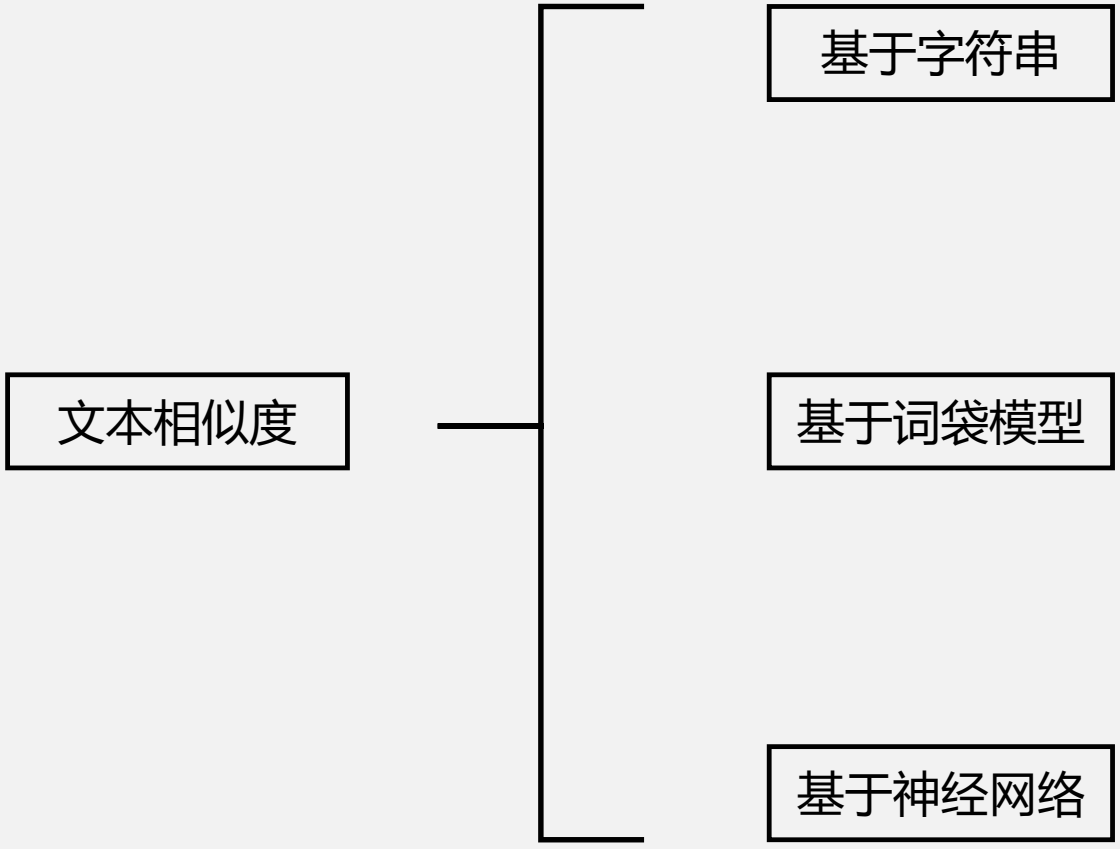
但相关度越大，不能说明相似度高

计算方法

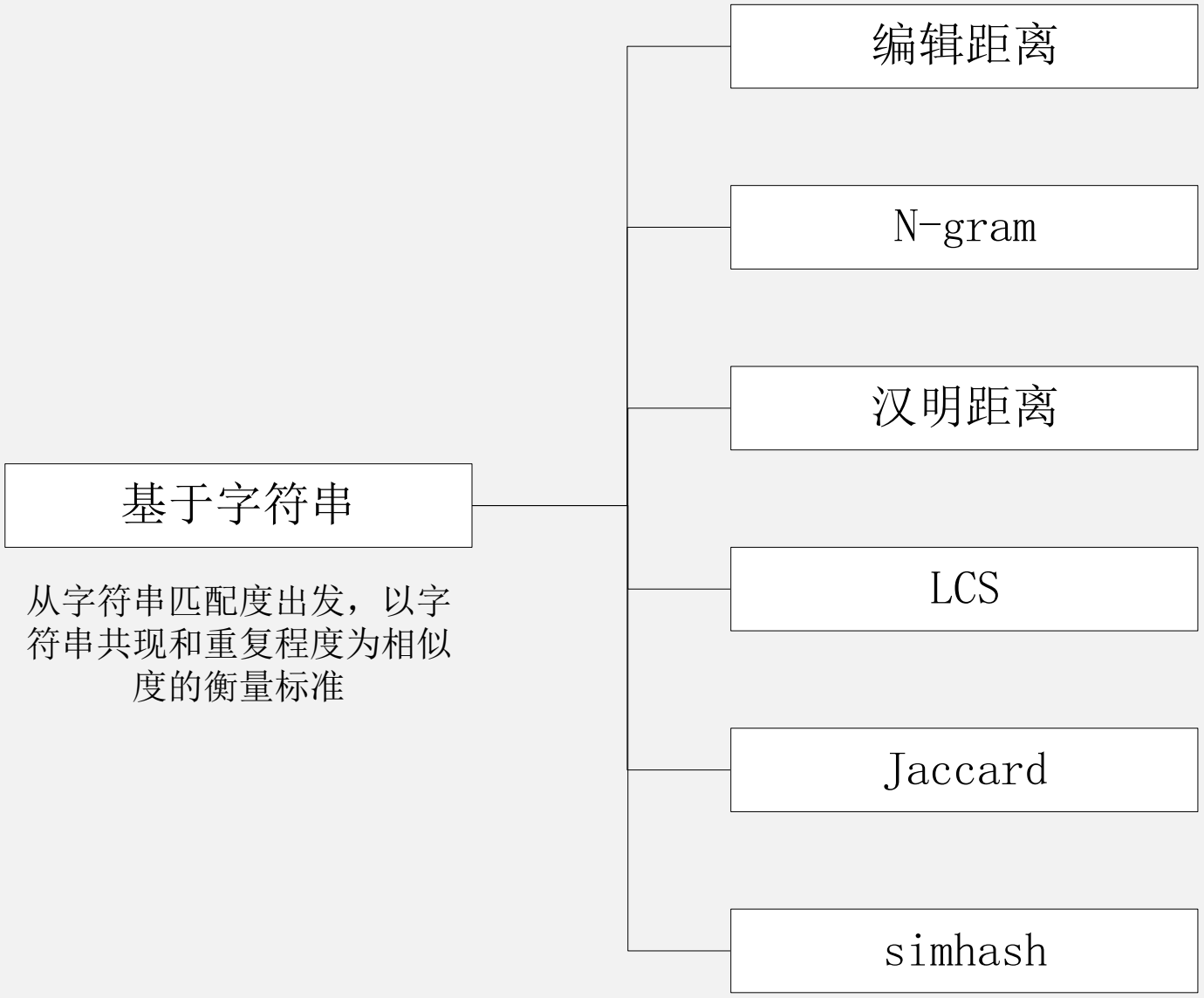
PART TWO



2 计算方法 | 计算方法分类



2.1 计算方法 | 基于字符串计算



2.1 基于字符串 | 编辑距离

S_A转换到S_B需要删除、插入、替换操作的最小次数

kitten->sitting:

kitten->sitten->sittin->sitting

D=3

DNA对比

纠错

2.1 基于字符串 | N-gram

按照N-Gram方法得到N个分词组成的子字符串，其中相同的子字符串个数作为N-Gram距离

S_A ="ABC"

(begin,A) 、 (A,B) 、 (B,C) 、 (C,end)

S_B ="AB"

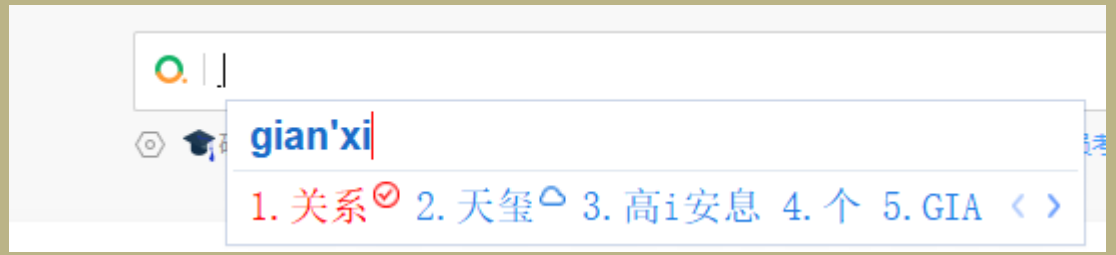
(begin,A) 、 (A,B) 、 (B,end)

N-gram=3

$$|G_N(s)| + |G_N(t)| - 2 \times |G_N(s) \cap G_N(t)|$$

| S_A | + | S_B | -2*N-gram

模糊匹配



S_A转换到S_B所需要替换的字符个数

D(1011101,1001001)=2

1 0 1 1 1 0 1

⊕

1 0 0 1 0 0 1

以图搜图

2.1 基于字符串 | LCS | Jaccard

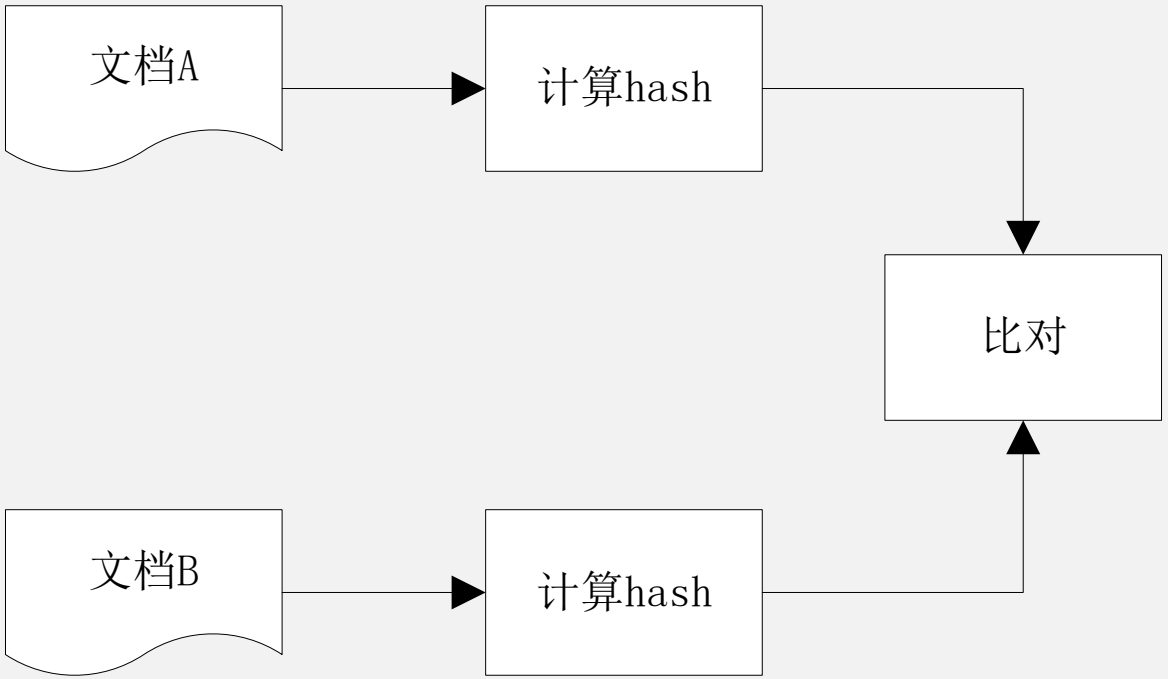
S_A，S_B两个字符串共现且最长的子字符串长度

D（12455，245576）=4

两个集合A和B的交集元素在A，B的并集中所占的比例

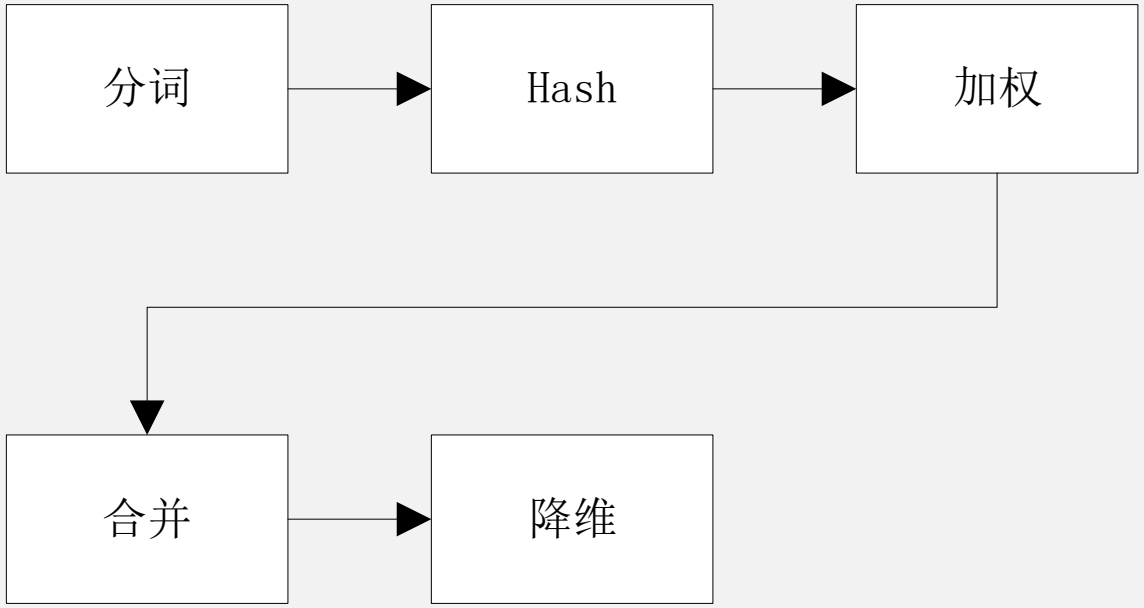
$$J(A,B)=\frac{|A\cap B|}{|A\cup B|}$$

传统hash

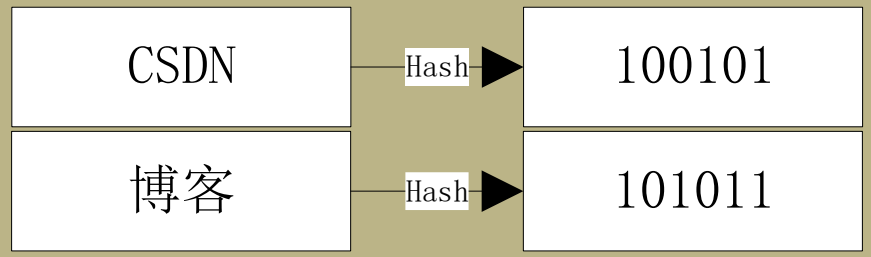


- irb(main):006:0> p1 = 'the cat sat on the mat'
  - irb(main):007:0> p1.hash => 415542861
- irb(main):005:0> p2 = 'the cat sat on a mat'
  - irb(main):007:0> p2.hash => 668720516
- irb(main):007:0> p3 = 'we all scream for ice cream'
  - irb(main):007:0> p3.hash => 767429688 "

simhash



listA=[‘这 ’, ‘只 ’, ‘皮靴 ’, ‘号码 ’, ‘大 ’, ‘了 ’, ‘那 ’, ‘只 ’, ‘号码 ’, ‘合适 ’]



Weight：4

W(CSDN)= 1 -1 -1 1 -1 1

×4

4 -4 -4 4 -4 4

W(CSDN) = 4 -4 -4 4 -4 4

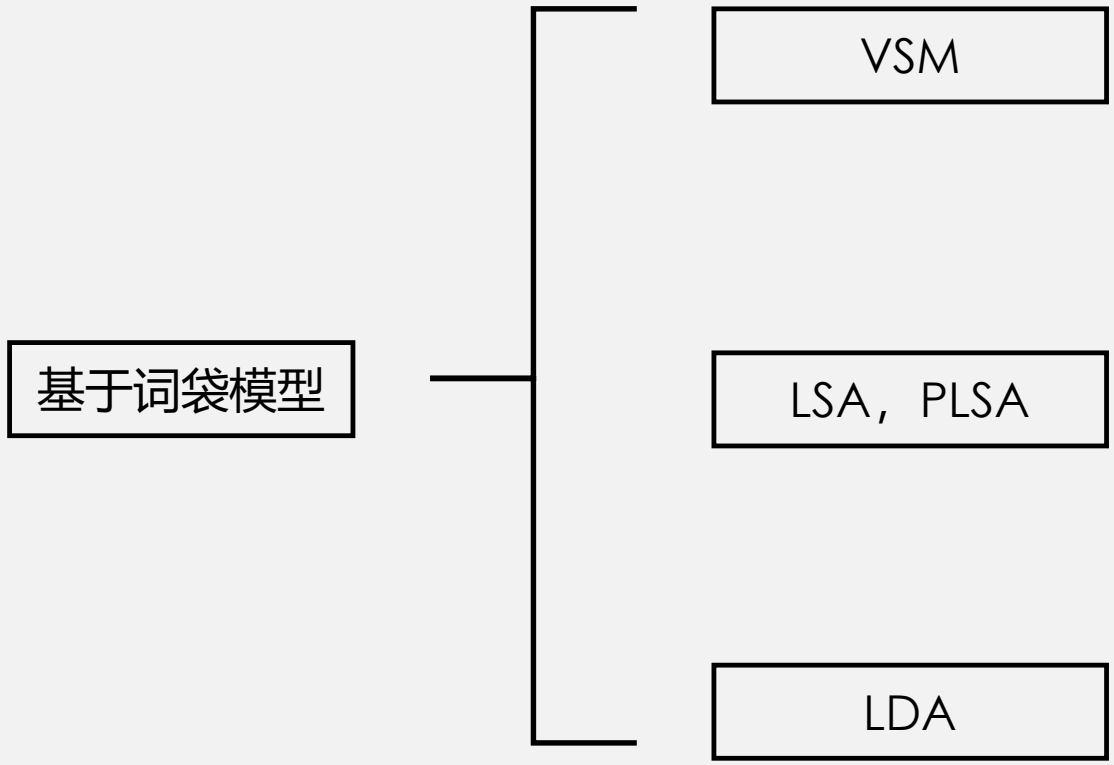
W(博客) = 5 -5 5 -5 5 5 +

9 -9 1 -1 1 9

降维结果： 1 0 1 0 1 1

2.1 基于字符串 | LCS | Jaccard | simhash

2.2 计算方法 | 计算方法分类



2.2 词袋模型| 相关方法

欧氏距离：两点之间的距离
N维上两点a（x1,x2..xn），b(y1,y2...yn)的欧式距离公式

$$d_{ab} = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + ... + (x_n - y_n)^2}$$

曼哈顿距离：也叫曼哈顿街区距离，就是开车从出发点到目的地的行驶距离
N维上两点a（x1,x2..xn），b(y1,y2...yn)的曼哈顿距离公式

$$d_{ab} = |x_1 - y_1| + |x_2 - y_2| + ... + |x_n - y_n|$$

这两种方法可以用相同的公式定义相似度：**s=1 / (1+d)**

2.2 词袋模型| 相关方法

余弦相似度

余弦公式：

$$\cos\Theta = \frac{a * b}{|a| * |b|}$$

N维上两点a（x1,x2..xn），b(y1,y2...yn)的夹角余弦公式：

$$\cos\Theta = \frac{x_1 * y_1 + x_2 * y_2 + \ldots + x_n * y_n}{\sqrt{x_1^2 + x_2^2 + \ldots + x_n^2} * \sqrt{y_1^2 + y_2^2 + \ldots + y_n^2}}$$

2.2 词袋模型| 相关方法

KL距离

$$D_{KL}(p,q)=\sum_{j=1}^{T} p_j \ln \frac{p_j}{q_j}$$

JS距离：

$$D_{js}(p,q)=\frac{1}{2}\left[D_{KL}\left(p,\frac{p+q}{2}\right)+D_{KL}\left(q,\frac{p+q}{2}\right)\right]$$

2.2 词袋模型 | VSM

向量空间模型的基本想法是：给定一个文本，用一个向量表示该文本的语义，向量的每一维对应一个单词，其数值是该单词在该文本中出现的频数或Tf-Idf。

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix}$$

优缺点：单词向量稀疏，计算效率高， 一词多义和一义多词

2.2 词袋模型 | LSA,PLSA

LSA基本思想：将文本从稀疏的高维词汇空间映射到低维的潜在语义空间，在潜在语义空间计算相似性。

LSA步骤：

1 建立单词-文档矩阵

2 对矩阵进行奇异值分解M=U*S*V

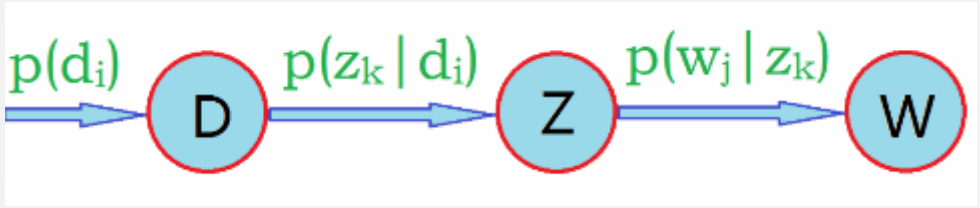
3 对分解后的矩阵进行降维：选择奇异值最大的t个数，同时保留矩阵U和V的前t列

4 使用降维后的矩阵重建单词-文档矩阵

优缺点：除噪，一义多词，缺乏统计学基础，SVD分解耗时

2.2 词袋模型 | LSA,PLSA

PLSA在LSA的基础上引入主题层，采用期望最大化（EM）算法训练主题，具备了统计基础，多义词和同义词在PLSA中分别被训练到不同的主题和相同的主题下。



(di ,wj)的联合分布为

$$p(d_i, w_j) = p(d_i)p(w_j|d_i)$$

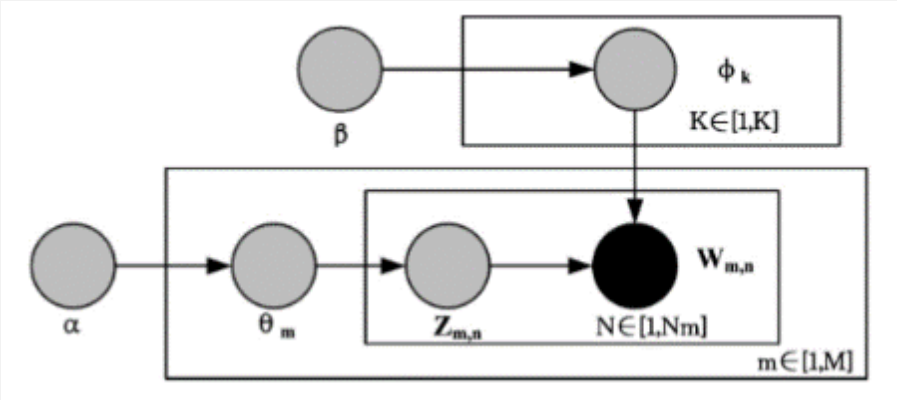
$$p(w_j|d_i) = \sum_{k=1}^{K} p(w_j|z_k)p(z_k|d_i).$$

优缺点： 避免了多义词、同义词的影响, 但不适用于大规模文本。

2.2 词袋模型 | LDA

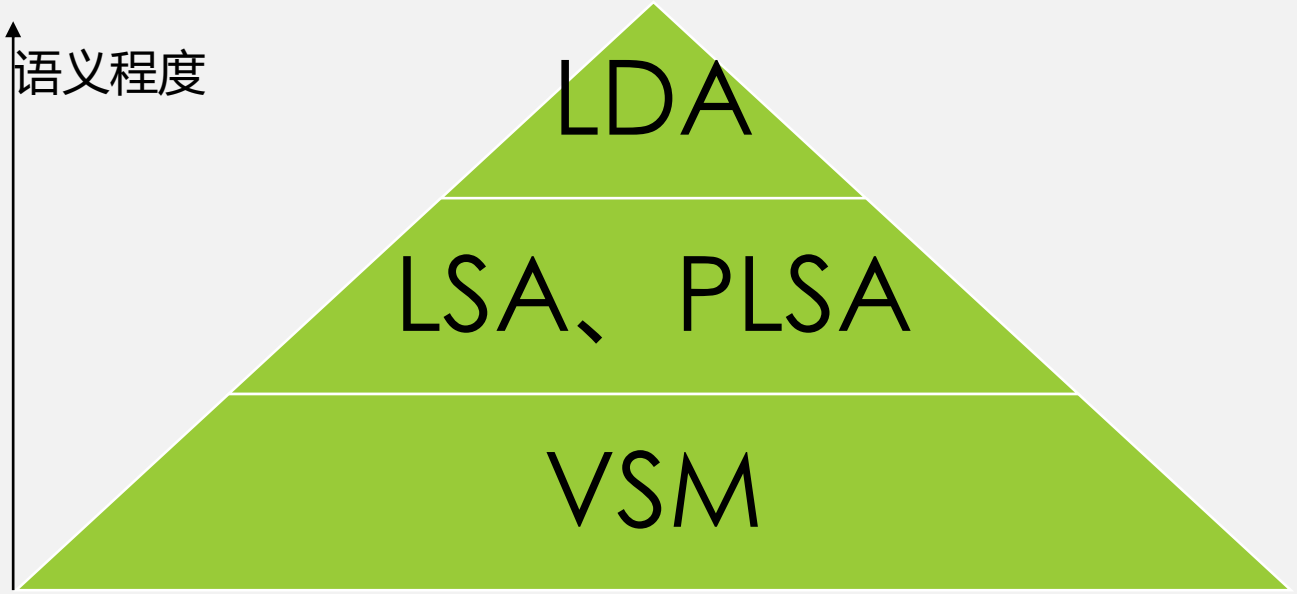
LDA主题模型是一个三层贝叶斯概率模型，包含词，主题和文档三层结构。

基本思想：对文本进行主题建模，并在主题对应的词语分布中遍历抽取文本中的词语，得到文本的主题分布，通过此分布计算文本相似度。LDA的文档到主题服从Dirichlet分布，主题到词服从多项式分布,。

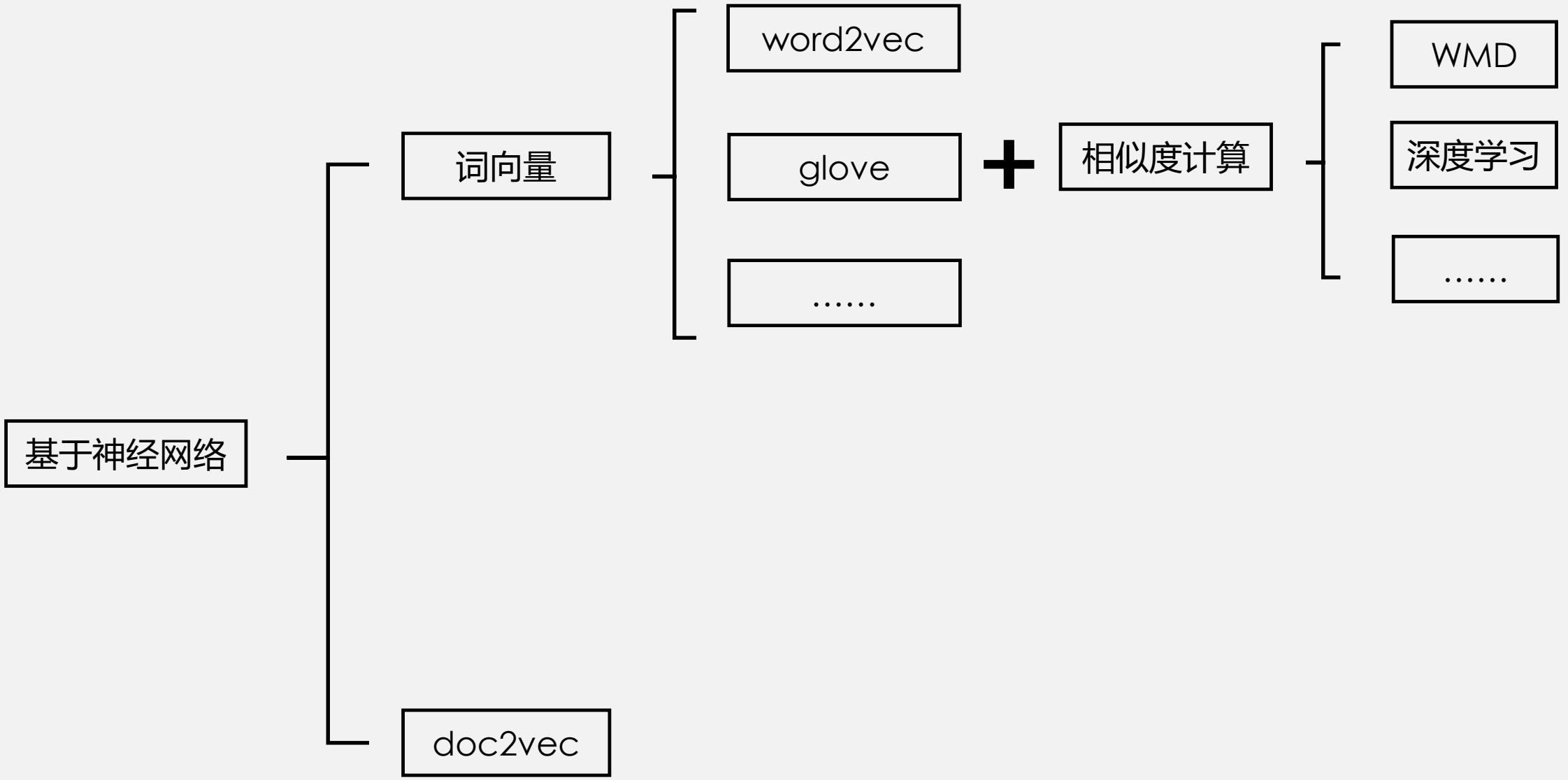


LDA模型的有向概率图

2.2 词袋模型 | 总结



2.3 计算方法 | 计算方法分类



2.3 神经网络 | Word2vec

Word2vec是一种由google提出的简单化的神经网络，用作将word在向量空间中进行表达。

Onehot编码是常用的单词表达形式，但是Onehot编码有几个缺点，维度高、稀疏、表达能力欠缺，而Word2vec正是用于弥补Onehot的这种缺点，利用将word嵌入到另一个实数向量空间从而完成降维，并且word的向量表示具备具体意义，向量的余弦距离通常也能刻画在预料中两个单词之间的差异性。

2.3 神经网络 | Word2vec

“i will go home and have my dinner.”

设中心词是home，取窗口为2，那么影响home单词的是will go和and have，我们希望P(home | (will,go,and,have))最大或者P((will,go,and,have) | home)最大，这两种不同的形式也就形成了Word2vec的两种形式

CBOW(Continuous Bag-of-Words Model)

根据窗口的上下词来预测当前的中心词

Skip-Gram(Continuous Skip-gram Model)

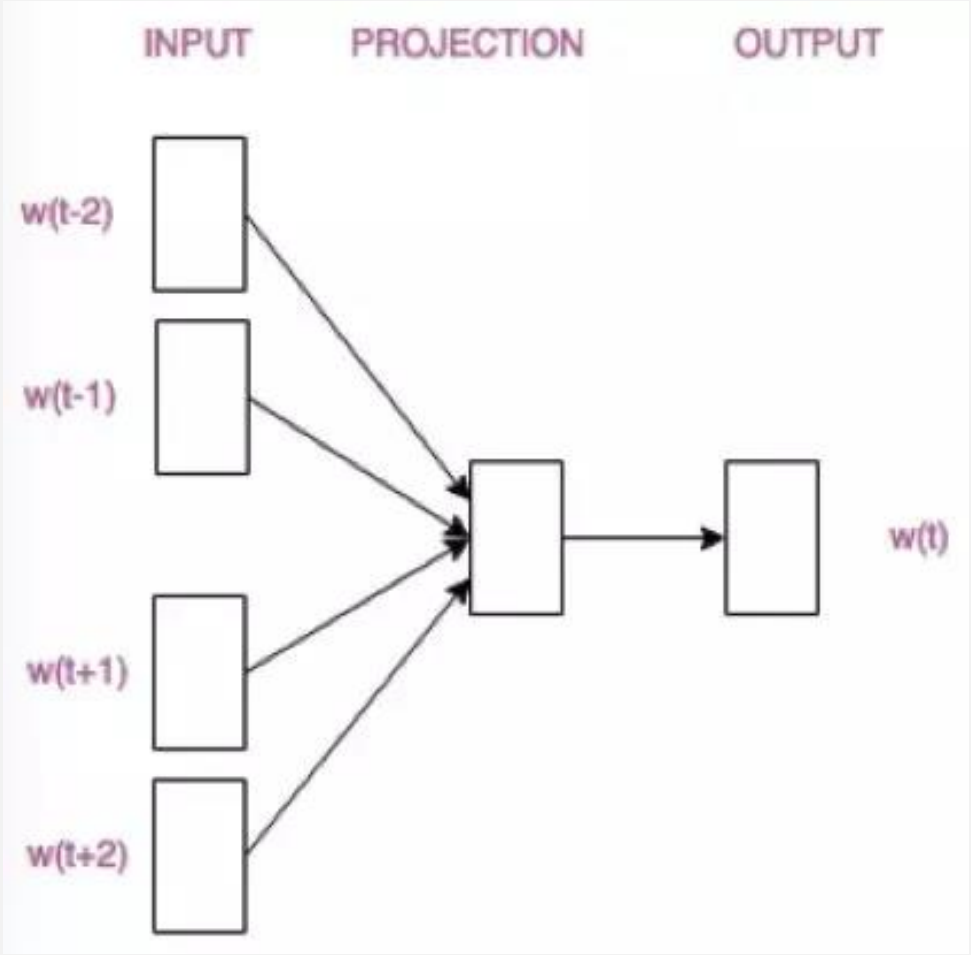
以当前的中心词来预测上下的窗口词

2.3 神经网络 | CBOW

CBOW模型的训练输入是某一个特征词的上下文相关的词对应的词向量，而输出就是这特定的一个词的词向量

如右图所示：窗口为2， $w_{(t)}$为特征词词向量，通过已知的当前值$w_{(t)}$的上下文$w_{(t-1)}$ $w_{(t-2)}$ $w_{(t+1)}$ $w_{(t+2)}$，来预测$w_{(t)}$

优化目标函数： $$L = \sum_{\omega \in C} logp(\omega | Context(\omega))$$



CBOW模型

2.3 神经网络 | CBOW

1 输入层：上下文单词的onehot.

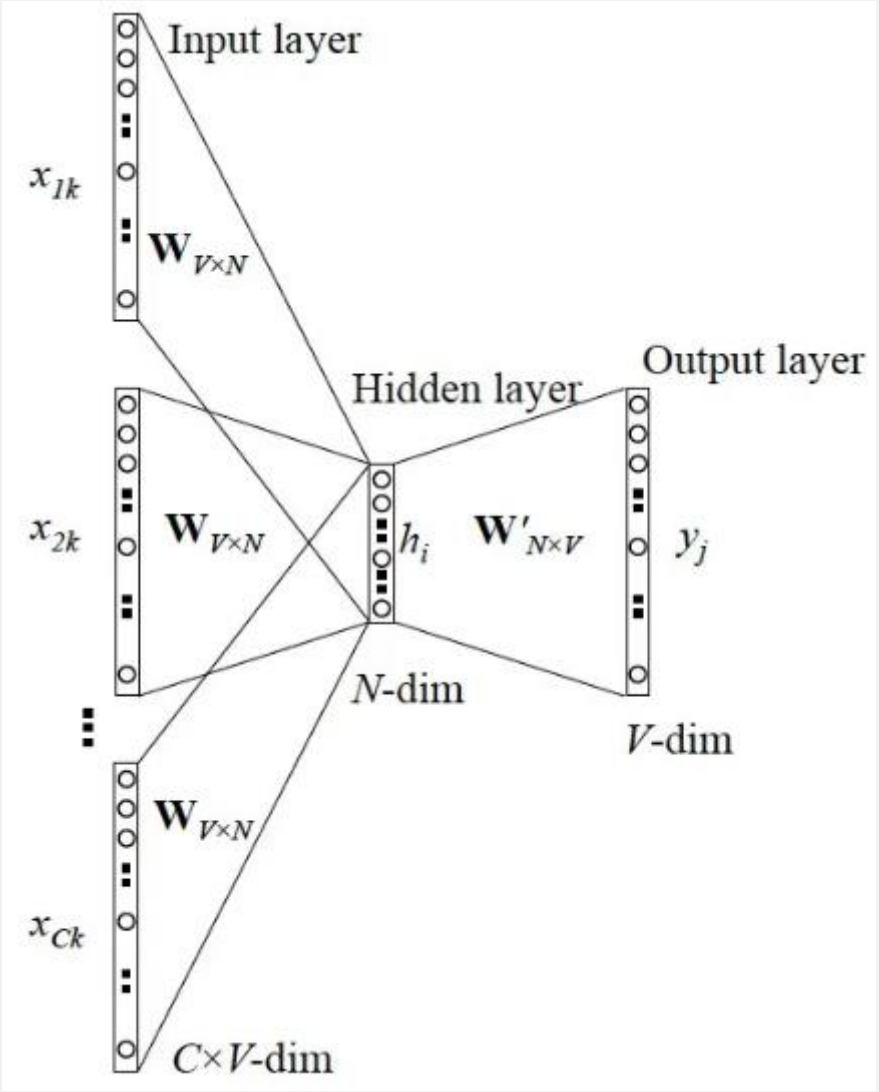
2 所有onehot分别乘以共享的输入权重矩阵W.

3 所得的相加求平均作为隐层向量, size为1N.

4 乘以输出权重矩阵W' {NV}

5 得到向量 {1V} 激活函数处理得到V-dim概率分布

6 概率最大的index所指示的单词为预测出的中间词（target word）与true label的onehot做比较，误差越小越好

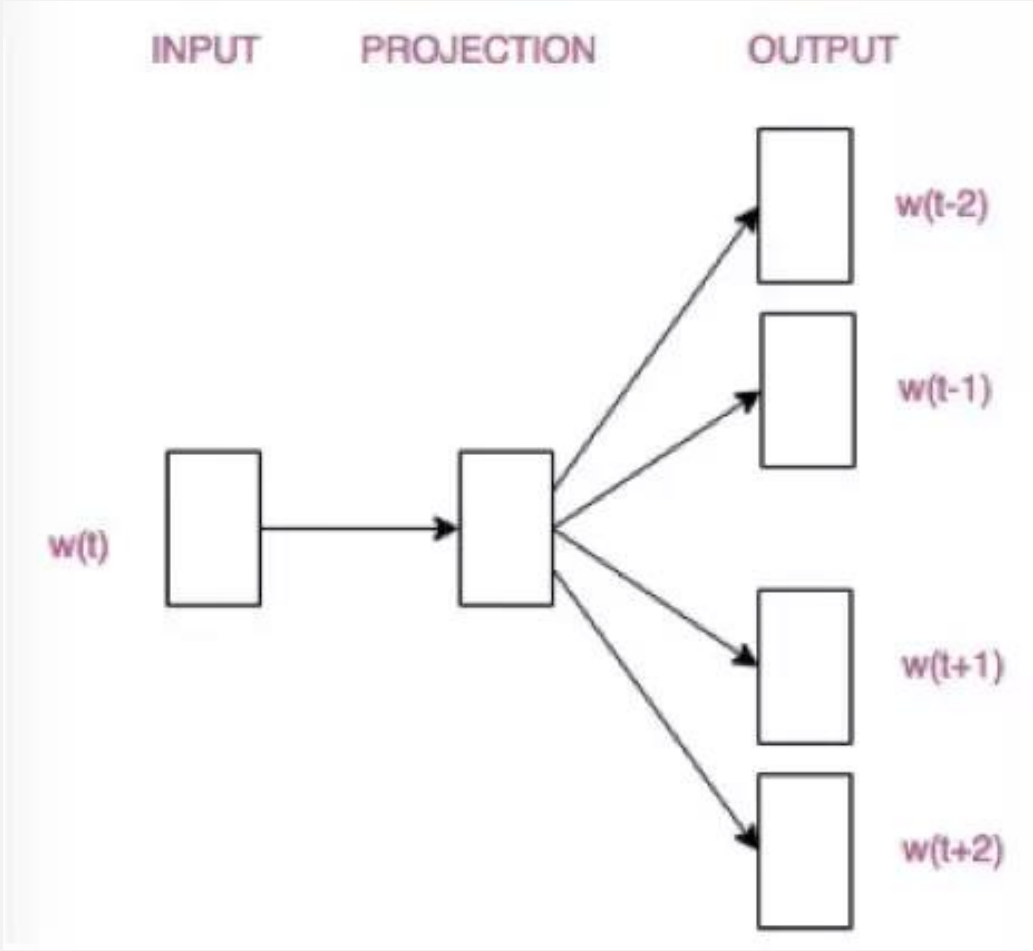


CBOW训练模型

2.3 神经网络 | Skip-Gram

如右图所示：窗口为2， $w_{(t)}$为特征词词向量，通过已知的当前值$w_{(t)}$，来预测$w_{(t)}$的上下文$w_{(t-1)}$ $w_{(t-2)}$ $w_{(t+1)}$ $w_{(t+2)}$

优化目标函数： $$L = \sum_{\omega \in C} logp(Context(\omega)|\omega)$$



Skip-Gram模型

2.3 神经网络 | Skip-Gram

参数：

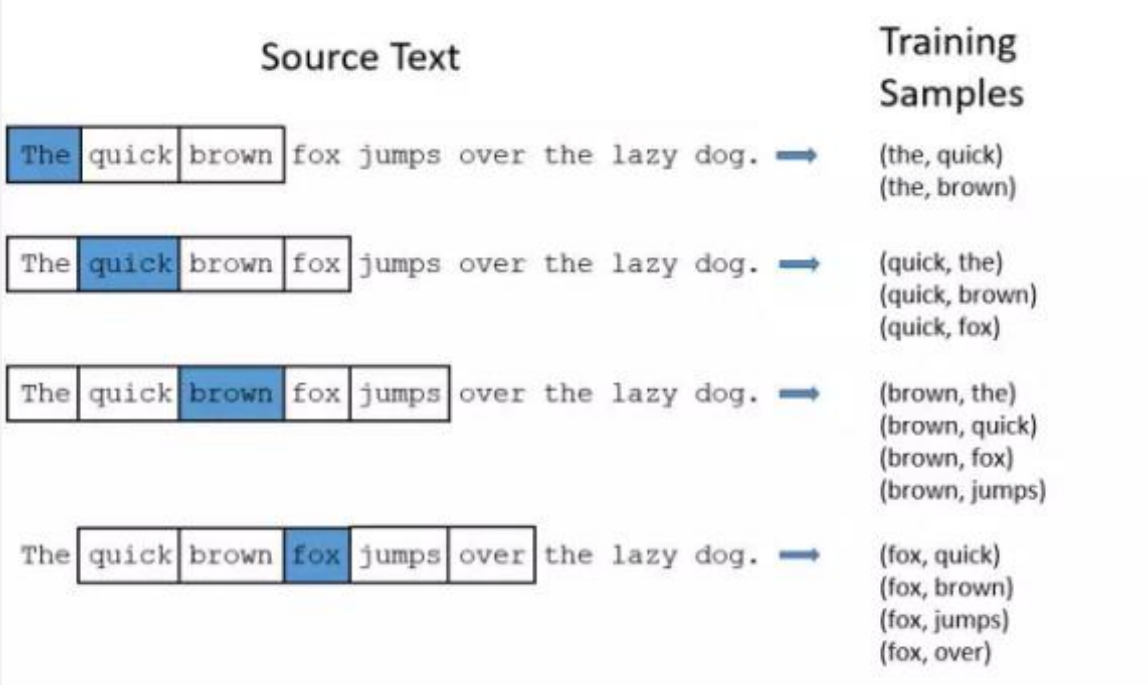
input word：输入，即关键词

skip_window：关键词左右词的数量

num_skips：从窗口中选取的不同次的数量

Output word：输入，某个关键词窗口内的上下文c

训练数据： num_skips 个(input word ,Output word)



Skip-Gram训练过程

2.3 神经网络 | GloVe模型

GloVe模型是一种词向量分布表示模型，是一种无监督学习算法，目标是通过进行词的向量化表示，使得向量之间尽可能多地蕴含语义和语法的信息。

总体上看，GloVe模型是一种对“词-词”矩阵进行分解从而得到词表示的方法

GloVe流程

2.3 神经网络 | GloVe模型

统计共现矩阵

设共现矩阵为X，其元素为$X_{i,j}$。
$X_{i,j}$的意义为：在整个语料库中，单词i和单词j共同出现在一个窗口中的次数。

如：存在语料库 i will go home and have my dinner，窗口宽度为5

窗口标号	中心词	窗口内容
0	i	i will go
1	will	i will go home
2	go	i will go home and
3	home	will go home and have
4	and	go home and have my
…	…	…

以窗口3为例，中心词为home，则$X_{home,will}$，$X_{home,go}$，$X_{home,and}$，$X_{home,have}$的值分别会+1，
使用窗口将整个语料库遍历一遍，即可得到共现矩阵X

2.3 神经网络 | GloVe模型

训练词向量

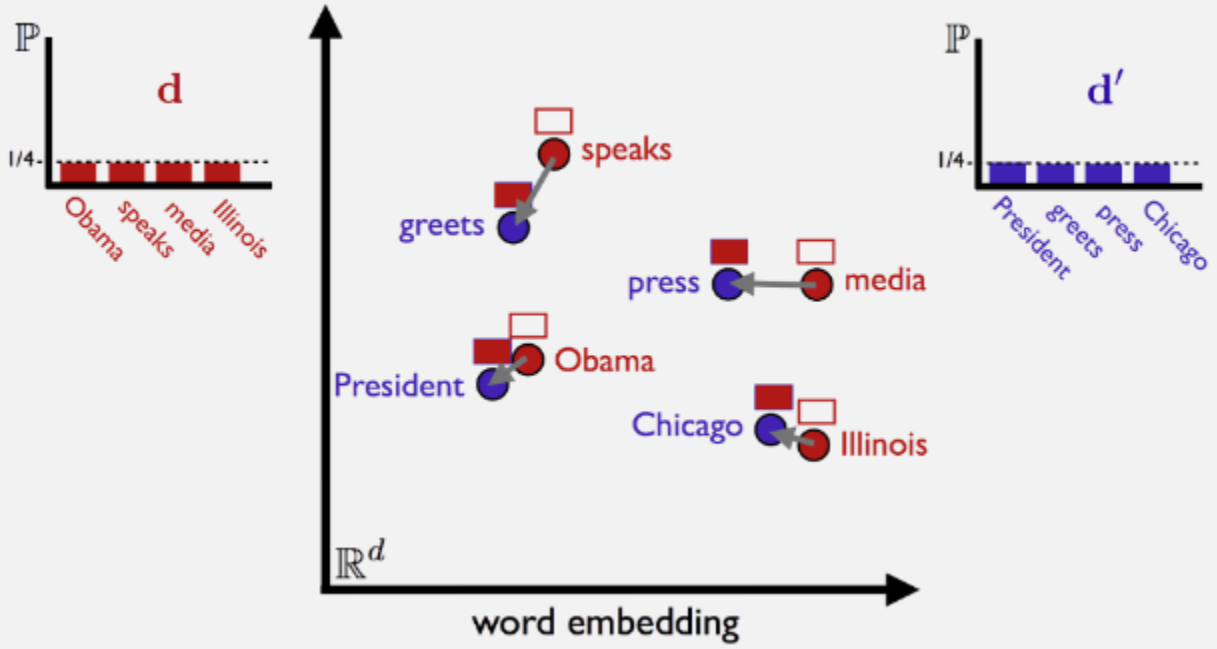
代价函数：

$$J = \sum_{i,j}^{N} f(X_{i,j})(v_i^T v_j + b_i + b_j - log(X_{i,j}))^2$$

v_i， v_j是单词i和单词j的词向量， b_i， b_j是两个标量（作者定义的偏差项），f是权重函数（具体函数公式及功能下一节介绍），N是词汇表的大小（共现矩阵维度为$N * N$）。

GloVe模型没有使用神经网络的方法

2.3 神经网络 | WMD



$$\sum_{i,j=1}^{n} T_{ij} \cdot c(i,j)$$

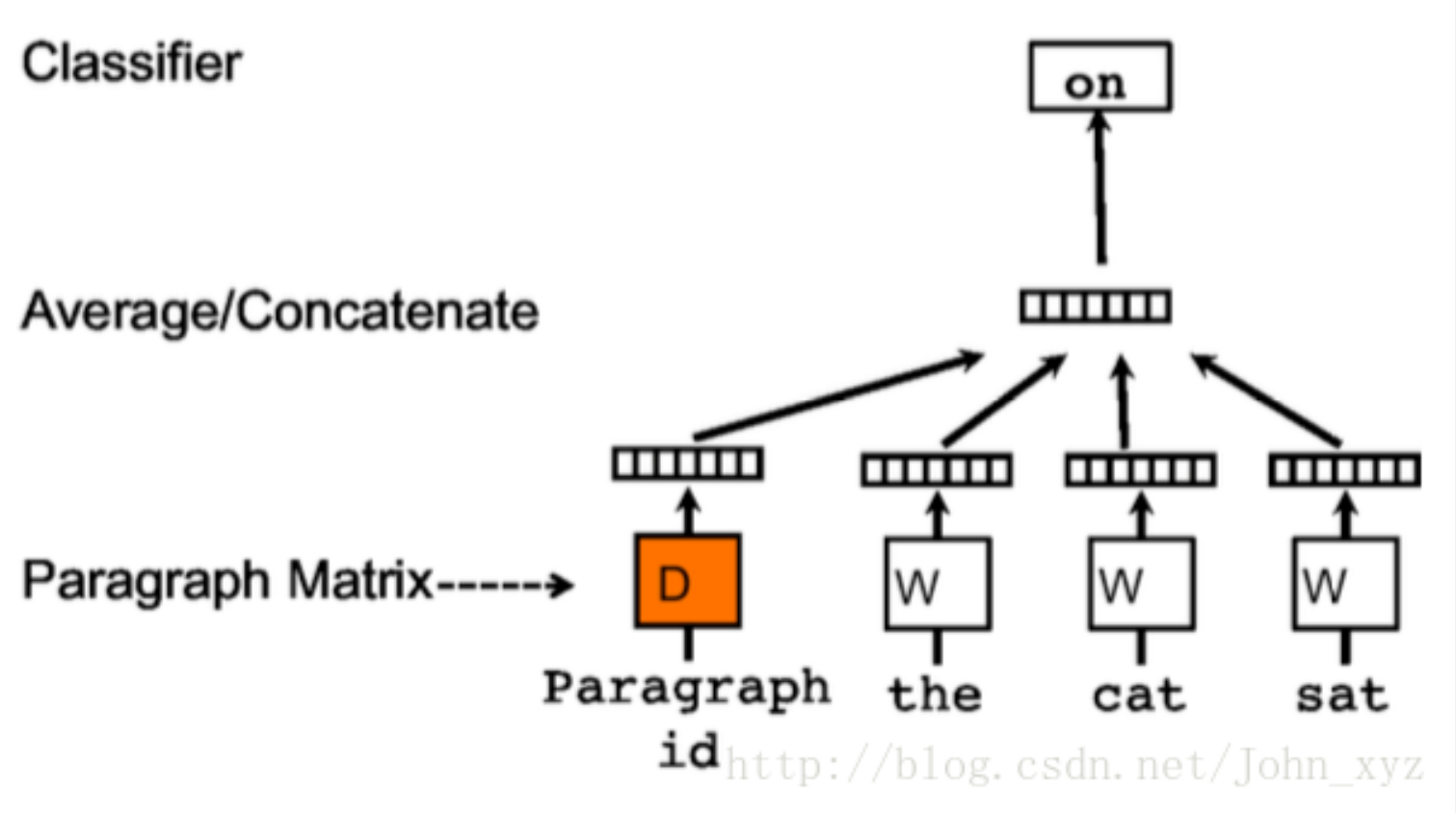
c(i,j)为i,j两个词所对应的词向量的欧氏距离，

Tij为相应的非负权重

2.3 神经网络 | doc2vec

- Doc2vec又叫Paragraph Vector是基于word2vec模型提出的，其具有一些优点，比如**不用固定句子长度，接受不同长度的句子做训练样本**
- Doc2vec是一个无监督学习算法，该算法用于预测一个向量来表示不同的文档，该模型的结构潜在的克服了词袋模型的缺点
- doc2vec也有两种训练方式，一种是PV-DM（Distributed Memory Model of paragraphvectors）类似于word2vec中的CBOW模型，另一种是PV-DBOW（Distributed Bag of Words of paragraph vector)类似于word2vec中的skip-gram模型

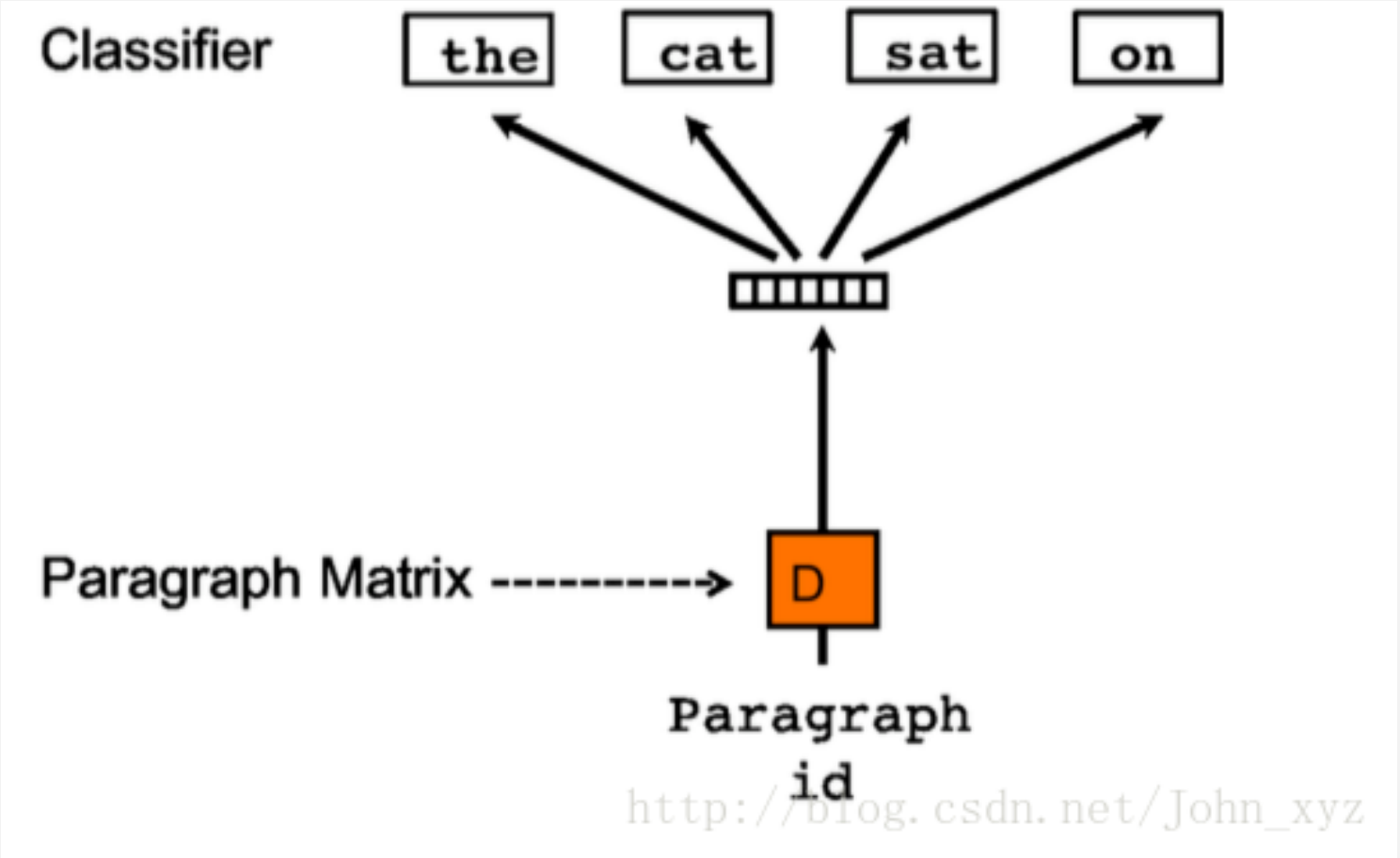
2.3 神经网络 | Doc2vec|PV-DM



输入词对应的词向量word vector和本句话对应的句子向量Paragraph vector作为输入层的输入

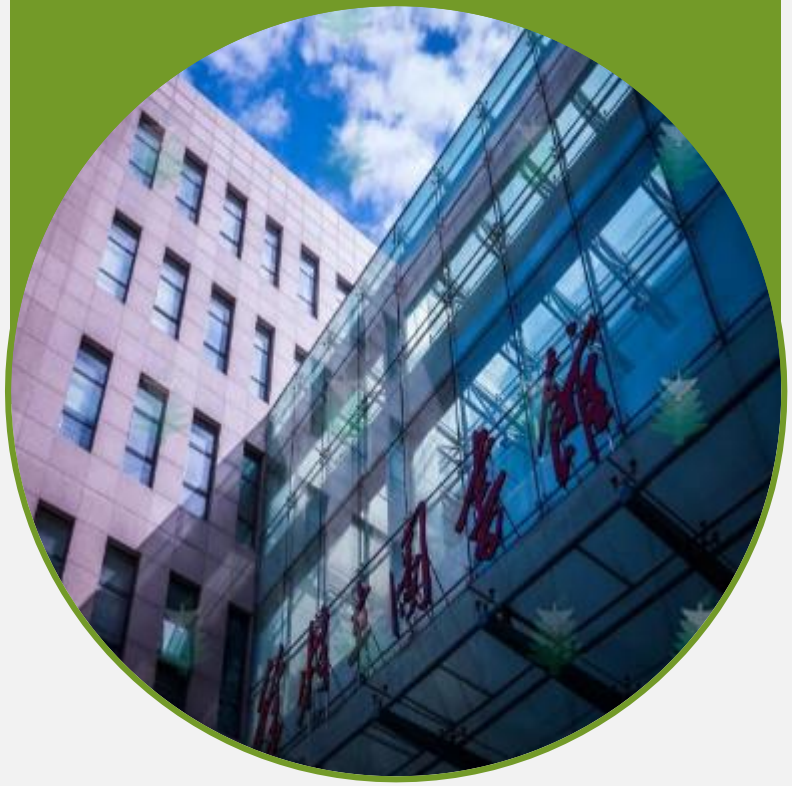
Paragraph Vector在同一个句子的若干次训练中是共享的

2.3 神经网络 | Doc2vec| PV-DBOW



实例演示

PART THREE



3.1 实例演示 | Gensim

- Gensim是一款开源的第三方Python工具包，用于从原始的非结构化的文本中，无监督地学习到文本隐层的主题向量表达。
- 它支持包括TF-IDF，LSA，LDA，和word2vec在内的多种主题模型算法

- 演示
  - Doc2vec
  - word2vec + WMD

3.1 实例演示 | Gensim

gensim.models.doc2vec.Doc2Vec(documents=None, dm_mean=None, dm=1, dbow_words=0, dm_concat=0, dm_tag_count=1, docvecs=None, docvecs_mapfile=None, comment=None, trim_rule=None, **kwargs)

- **vector_size**：输出特征向量维度
- **Dm**：训练算法：默认为1，指**DM**；dm=0,则使用DBOW
- **window**：窗口大小，表示当前词与预测词在一个句子中的最大距离是多少
- **alpha:** 是初始的学习速率，在训练过程中会线性地递减到min_alpha
- **min_count:** 可以对字典做截断. 词频少于min_count次数的单词会被丢弃掉, 默认值为5
- **workers**：用于控制训练的并行数

3.1 实例演示 | Gensim

gensim.models.word2vec.Word2Vec(sentences=None, corpus_file=None, size=100, alpha=0.025, window=5, min_count=5, max_vocab_size=None, sample=0.001, seed=1, workers=3, min_alpha=0.0001, sg=0, hs=0, negative=5, ns_exponent=0.75, cbow_mean=1, iter=5, null_word=0, trim_rule=None, sorted_vocab=1, batch_words=10000)

- **Sentences** 供训练的句子，可以使用简单的列表
- **size** (int, optional) – word向量的维度
- **window** (int, optional) – 一个句子中当前单词和被预测单词的最大距离
- **min_count** (int, optional) – 忽略词频小于此值的单词
- **sg** 模型的训练算法: 1: skip-gram; 0: **CBOW**.
- **alpha** 是初始的学习速率，在训练过程中会线性地递减到min_alpha
- **workers** 用于控制训练的并行数

3.1 实例演示 | Gensim

KeyedVectors.wmdistance(self, document1, document2) #计算词移距离

3.1 实例演示 | 语料库

- 来源：斯坦福大学自然语言语料库
- 数量：367373条记录
- 格式：句子1, 句子2，0表示不相似，1表示相似

```
A person on a horse jumps over a broken down airplane.  A person is at a diner, ordering an omelette.  0
A person on a horse jumps over a broken down airplane.  A person is outdoors, on a horse.  1
Children smiling and waving at camera  There are children present  1
Children smiling and waving at camera  The kids are frowning  0
A boy is jumping on skateboard in the middle of a red bridge.  The boy skates down the sidewalk.  0
A boy is jumping on skateboard in the middle of a red bridge.  The boy does a skateboarding trick.  1
```

3.1 实例演示 | doc2vec

```
def getDataSets(filepath):
    x1,x2,y = getTsvData(filepath)
    np.random.seed(131)
    shuffle_indices = np.random.permutation(np.arange(len(y)))
    x1_shuffled = x1[shuffle_indices]
    x2_shuffled = x2[shuffle_indices]
    y_shuffled = y[shuffle_indices]
    dev_idx = -1 * len(y_shuffled) * percent_dev // 100
    del x1
    del x2
    x1_train, x1_dev = x1_shuffled[:dev_idx], x1_shuffled[dev_idx:]
    x2_train, x2_dev = x2_shuffled[:dev_idx], x2_shuffled[dev_idx:]
    y_train, y_dev = y_shuffled[:dev_idx], y_shuffled[dev_idx:]
    return [list(x1_train),list(x2_train),list(y_train)],[list(x1_dev),list(x2_dev),list(y_dev)]
```

加载数据集，去除标点符号和停用词，8:2划分训练集和测试集

3.1 实例演示 | doc2vec

```
def train(data):
    print(data)
    labels = range(len(data))
    sentences = LabeledLineSentence(data, labels)
    model = gensim.models.Doc2Vec(vector_size=400, window=15, min_count=5,
                                  workers=4, alpha=0.025, min_alpha=0.025, epochs=25, sample=1e-5)
    model.build_vocab(sentences)
    print("开始训练...")
    model.train(sentences, total_examples=model.corpus_count, epochs=12)

    model.save("./doc2vec.model")
    print("model saved")
```

训练doc2vec模型，用gensim.models.doc2vec.TaggedDocument() 为文档打tag

3.1 实例演示 | doc2vec

```
def similarity(a_vect, b_vect):
    dis = 0
    for a, b in zip(a_vect, b_vect):
        dis+= (a-b)**2
    dis = dis**0.5
    dis = 1/(1+dis)
    return dis
```

得到句子向量后使用欧式距离计算相似度，并归一化

3.1 实例演示 | doc2vec

标签	相似度	标签	相似度
1	0.7190369983186621	0	0.656975968822355 1
1	0.7285067546157017	0	0.656975968822355 1
0	0.6709738815201839	0	0.617431376344419 8
0	0.6468395283662252	0	0.543137362799526 3
1	0.4827822341061124	1	0.67657212072385 3
0	0.4536953395109398	0	0.605364801233434 1
0	0.6885245310819658	1	0.770360458895408 7
1	0.6146827604414465	1	0.679200712725213 1
0	0.6967068465749284	1	0.677073614472755 3
1	0.6163320152846937	0	0.553002960670659 2
1	0.663699879040724	0	0.510904191752271 6

3.1 实例演示 | doc2vec

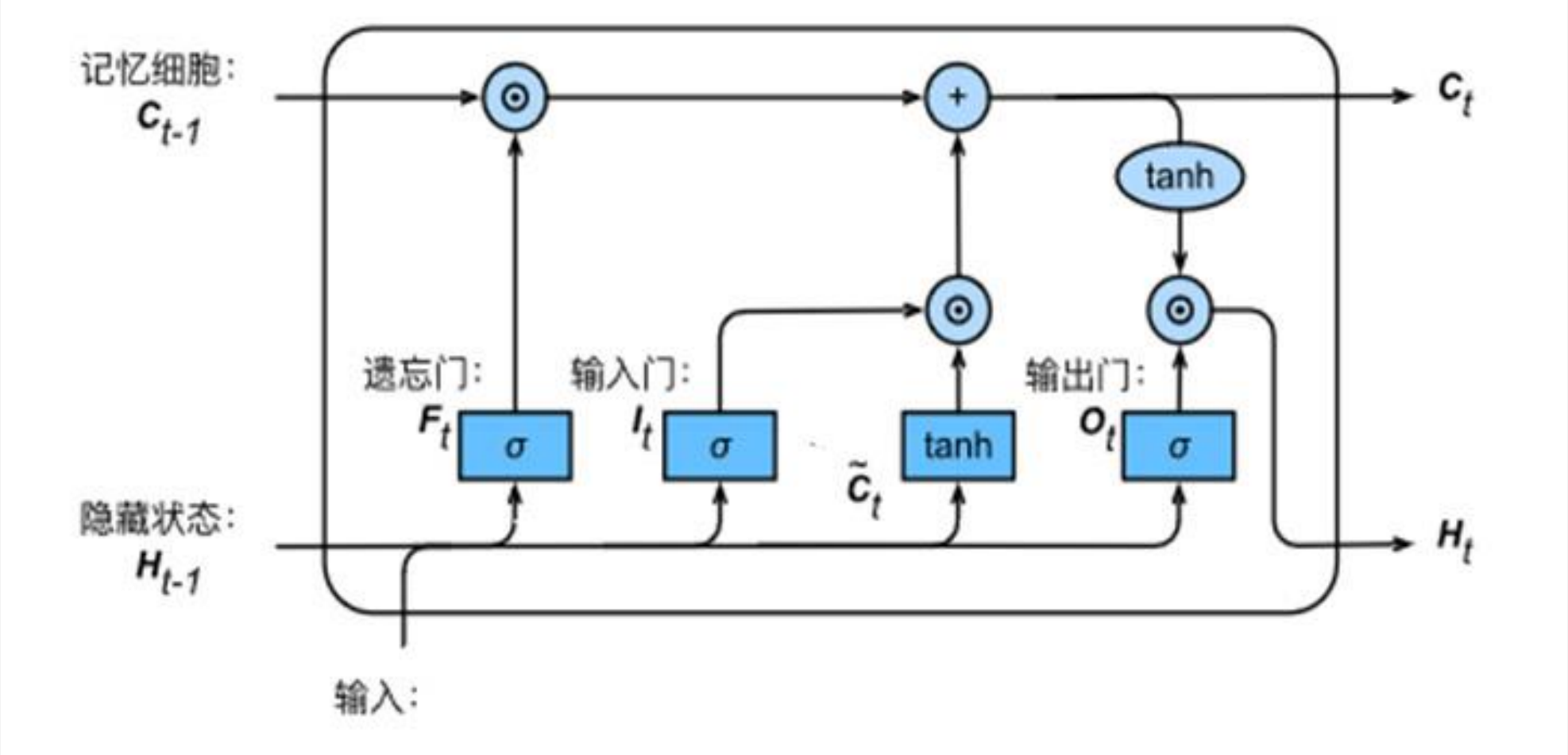
```
clf = svm.SVC(kernel='rbf')
result = []
for i in range(5):
    x_train, x_test, y_train, y_test = \
        model_selection.train_test_split(x, y, test_size=0.2)
    clf.fit(x_train, y_train)
    result.append(np.mean(y_test == clf.predict(x_test)))
print("svm classifier accuacy:")
print(result)
```

使用svm进行监督学习分类

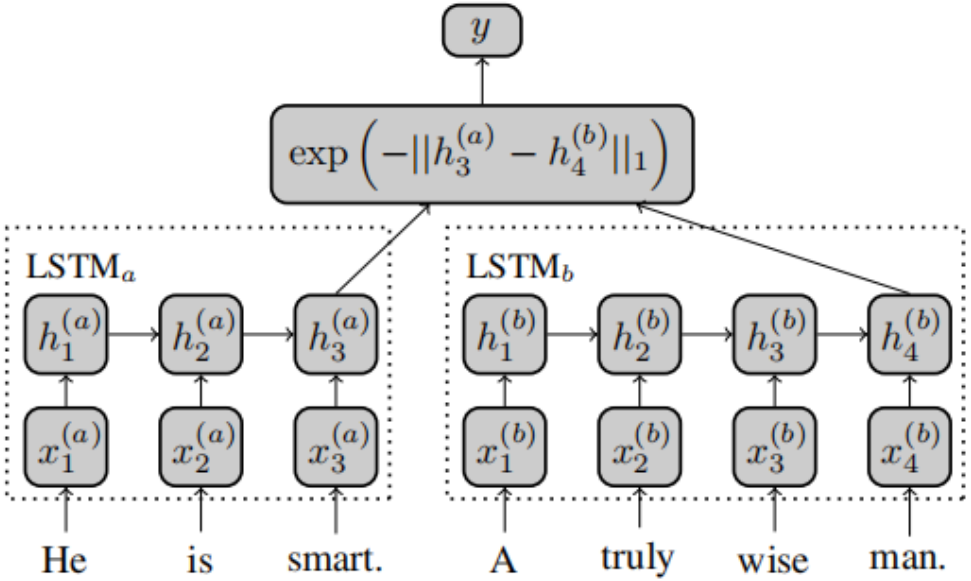
3.1 实例演示 | Result

方法	1	2	3	4	5
doc2vec	0.5533	0.5510	0.5509	0.5487	0.5508
Word2vec+WMD	0.6604	0.6541	0.6500	0.6507	0.6552
GoogleNews-vectors + WMD	0.6552	0.6469	0.6526	0.6492	0.6565

3.2 实例演示 | LSTM


记忆细胞:
C_{t-1}
C_t
tanh
遗忘门:
F_t
σ
输入门:
I_t
σ
$\tilde{C}_t$
tanh
输出门:
O_t
σ
隐藏状态:
H_{t-1}
H_t
输入:

3.2 实例演示 | 孪生网络



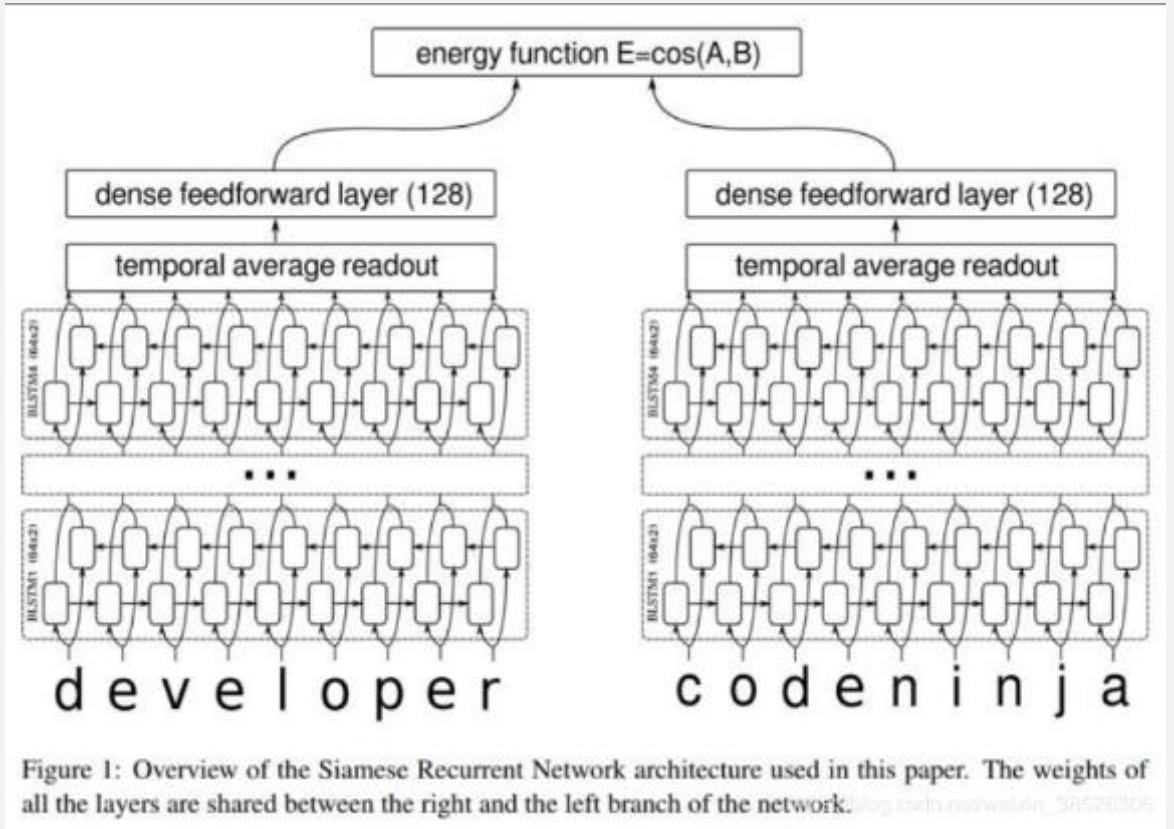


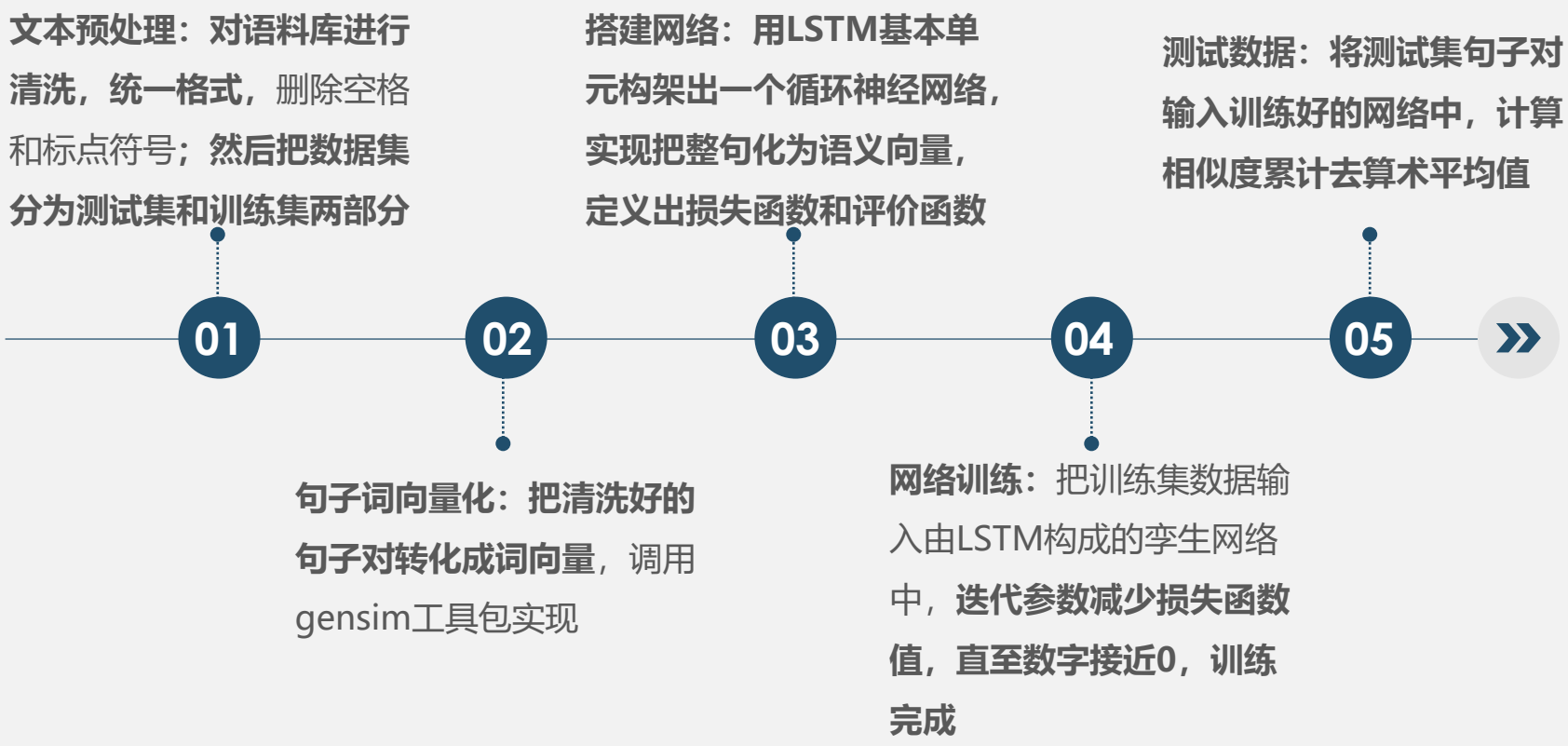
Figure 1: Overview of the Siamese Recurrent Network architecture used in this paper. The weights of all the layers are shared between the right and the left branch of the network.

3.2 实例演示 | 训练数据集

Stanford Natural Language Inference (SNLI)
斯坦福大学自然语言语料库

```
Children smiling and waving at camera    There are children present  1
Children smiling and waving at camera    The kids are frowning    0
A boy is jumping on skateboard in the middle of a red bridge.    The boy skates down the sidewalk.    0
A boy is jumping on skateboard in the middle of a red bridge.    The boy does a skateboarding trick.  1
An older man sits with his orange juice at a small table in a coffee shop while employees in bright colored shirts smile in the background.
Two blond women are hugging one another.     The women are sleeping.  0
Two blond women are hugging one another.     There are women showing affection.   1
A few people in a restaurant setting, one of them is drinking orange juice.  The people are sitting at desks in school.   0
A few people in a restaurant setting, one of them is drinking orange juice.  The diners are at a restaurant.  1
An older man is drinking orange juice at a restaurant.   A man is drinking juice.     1
An older man is drinking orange juice at a restaurant.   Two women are at a restaurant drinking wine.     0
A man with blond-hair, and a brown shirt drinking out of a public water fountain.    A blond man wearing a brown shirt is reading a book on a
A man with blond-hair, and a brown shirt drinking out of a public water fountain.    A blond man drinking water from a fountain.  1
Two women who just had lunch hugging and saying goodbye.     The friends scowl at each other over a full dinner table.    0
Two women who just had lunch hugging and saying goodbye.     There are two woman in this picture.     1
Two women, holding food carryout containers, hug.    Two groups of rival gang members flipped each other off.     0
Two women, holding food carryout containers, hug.    Two women hug each other.    1
```

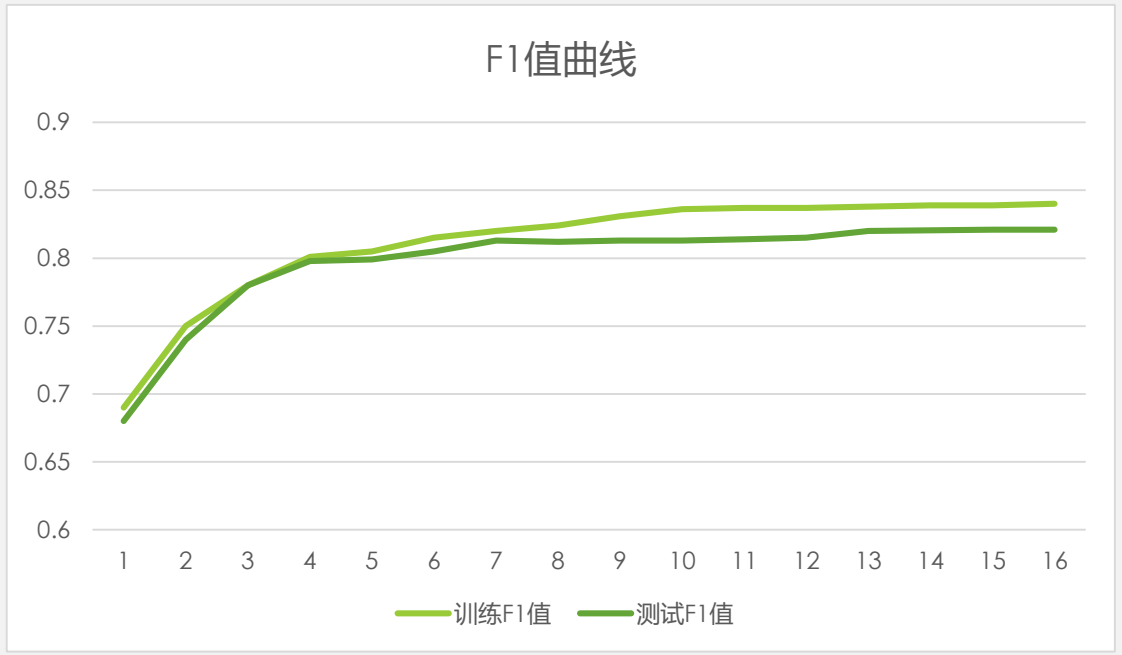
3.2 实例演示 | 程序流程

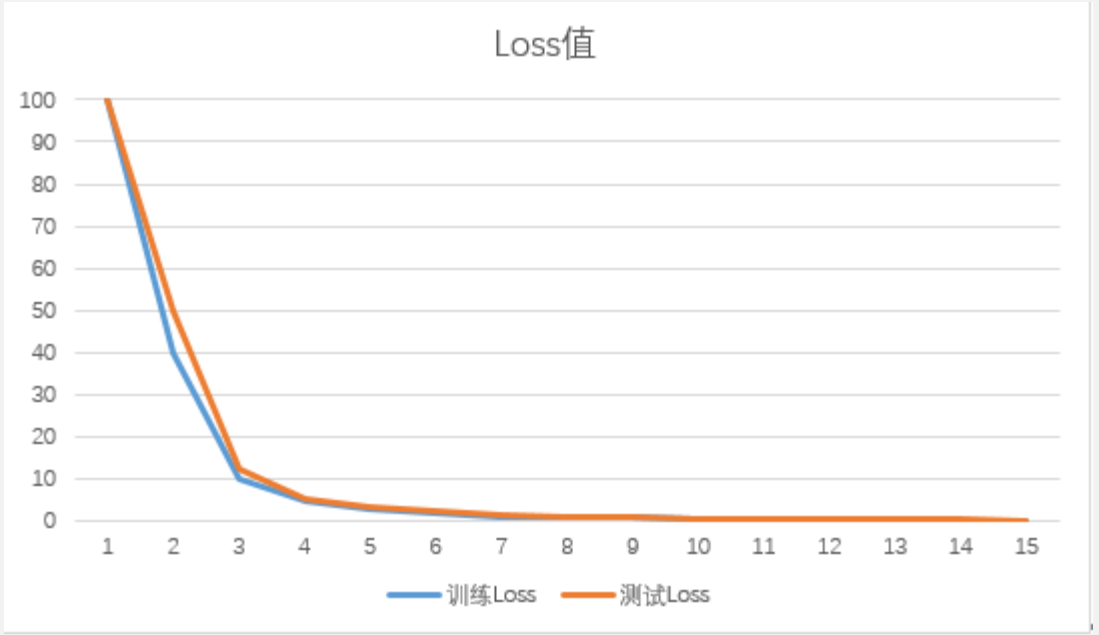

文本预处理：对语料库进行清洗，统一格式，删除空格和标点符号；然后把数据集分为测试集和训练集两部分
01
句子词向量化：把清洗好的句子对转化成词向量，调用gensim工具包实现
02
搭建网络：用LSTM基本单元构架出一个循环神经网络，实现把整句化为语义向量，定义出损失函数和评价函数
03
网络训练：把训练集数据输入由LSTM构成的孪生网络中，迭代参数减少损失函数值，直至数字接近0，训练完成
04
测试数据：将测试集句子对输入训练好的网络中，计算相似度累计去算术平均值
05

3.2 实例演示 | 模型参数

```
tf.flags.DEFINE_boolean("is_char_based", True, "is character based syntactic similarity. "
                                               "if false then word embedding based semantic similarity is used."
                                               "(default: True)")

tf.flags.DEFINE_string("word2vec_model", "wiki.simple.vec", "word2vec pre-trained embeddings file (default: None)")
tf.flags.DEFINE_string("word2vec_format", "text",
                       "word2vec pre-trained embeddings file format (bin/text/textgz)(default: None)")

tf.flags.DEFINE_integer("embedding_dim", 300, "Dimensionality of character embedding (default: 300)")
tf.flags.DEFINE_float("dropout_keep_prob", 1.0, "Dropout keep probability (default: 1.0)")
tf.flags.DEFINE_float("l2_reg_lambda", 0.0, "L2 regularizaion lambda (default: 0.0)")
tf.flags.DEFINE_string("training_files", "train_snli.txt",
                       "training file (default: None)")  # for sentence semantic similarity use "train_snli.txt"
tf.flags.DEFINE_integer("hidden_units", 50, "Number of hidden units (default:50)")

# Training parameters
tf.flags.DEFINE_integer("batch_size", 64, "Batch Size (default: 64)")
tf.flags.DEFINE_integer("num_epochs", 300, "Number of training epochs (default: 200)")
tf.flags.DEFINE_integer("evaluate_every", 1000, "Evaluate model on dev set after this many steps (default: 100)")
tf.flags.DEFINE_integer("checkpoint_every", 1000, "Save model after this many steps (default: 100)")
# Misc Parameters
tf.flags.DEFINE_boolean("allow_soft_placement", True, "Allow device soft device placement")
tf.flags.DEFINE_boolean("log_device_placement", False, "Log placement of ops on devices")
```

3.2 实例演示 | 实验结果






```
Accuracy: 0.804259
```

Thanks!