后厂理工学院
houchangtech.com

HCT NLP Week 2

问答摘要与推理
词向量实践

Outline

1. 词向量计算优化方法
2. 初识深度学习框架
3. Genism代码实践
4. 作业

3,23

python

n

log

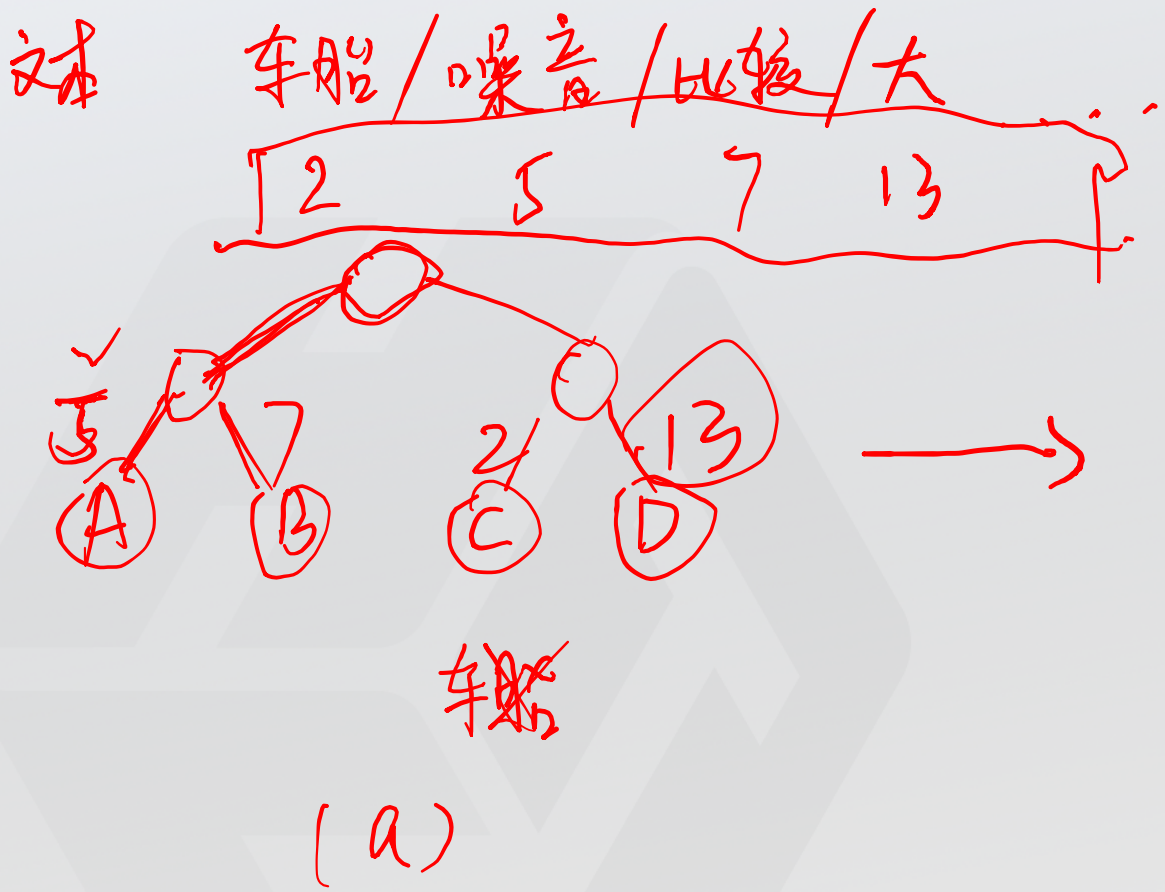
sorted 分组

1. 词向量计算优化方法

O(v) → O(logV)

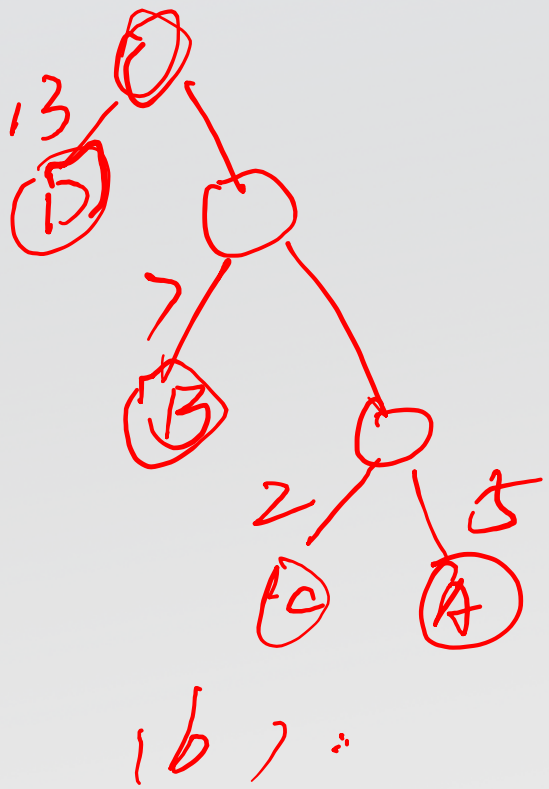
Hierarchical Softmax

Huffman Tree（哈夫曼树）



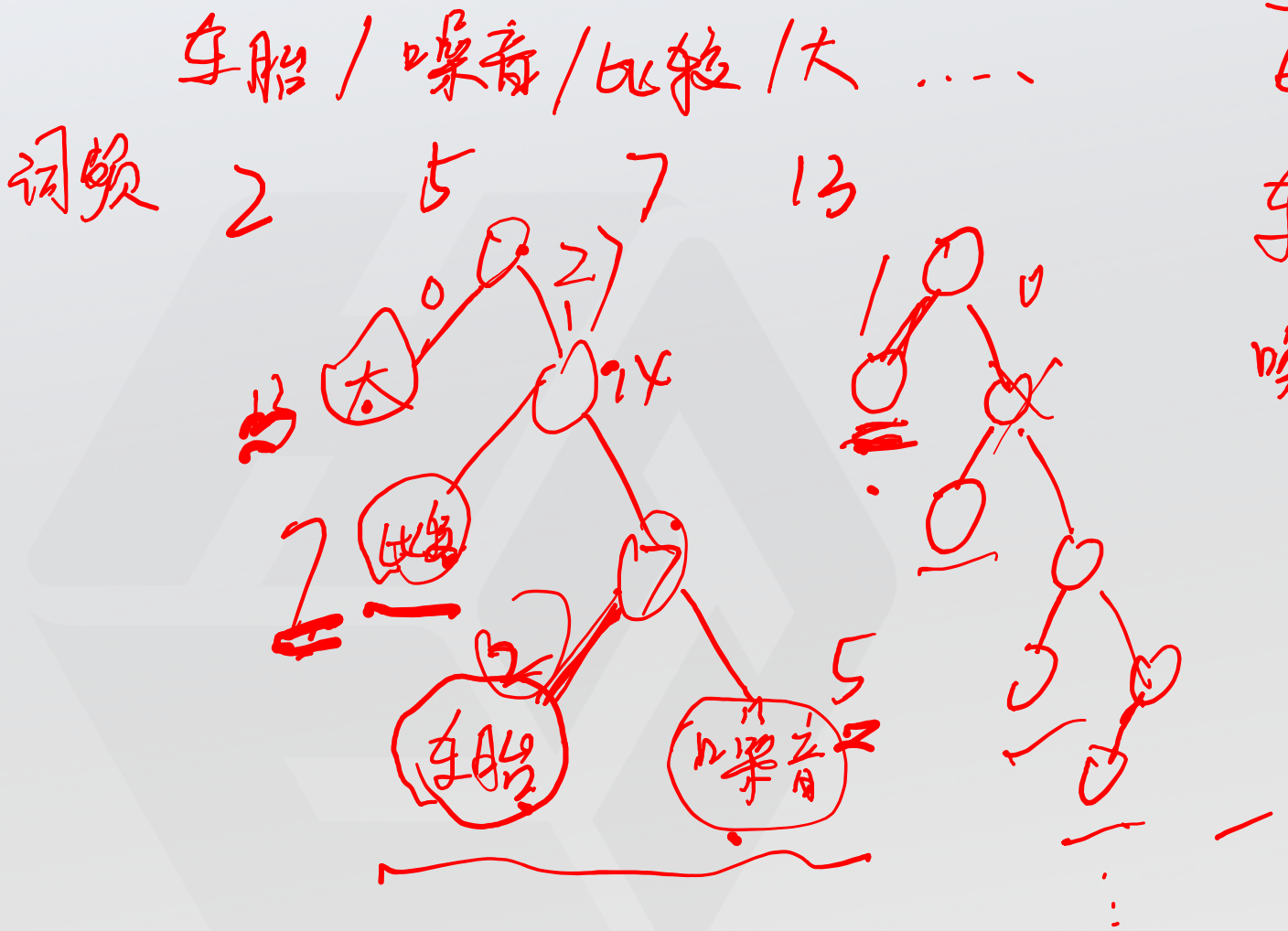
$Path\_a = 5\times2 + 7\times2 + 2\times2 + 13\times2 = 54$

$Path\_b = 13\times1 + 7\times2 + 2\times3 + 5\times3 = 48$



Hierarchical Softmax

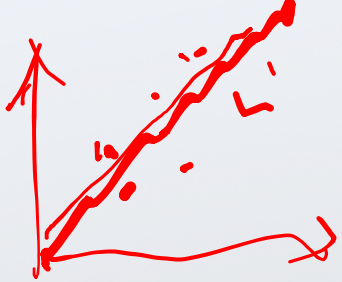
Huffman Tree（哈夫曼树）



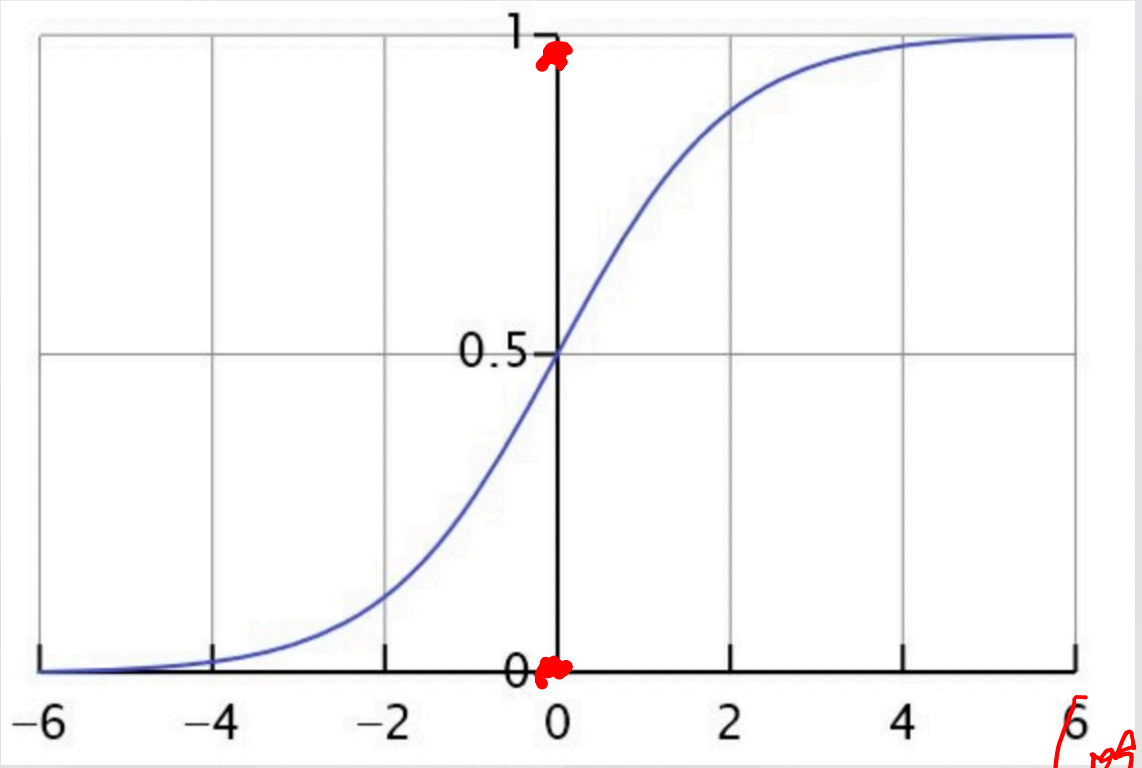
编码
大：0
蚊子：10
车胎：110
噪音：111

1) V个叶子节点
V-1个中间节点
2) ~~对应
3) ~~

Hierarchical Softmax



Logistic Regression LR



$\hat{y} = P(y=1|x) \quad 0 \le \hat{y} \le 1$

$\hat{y} = w^T x + b$

$w^T x = \sum_{i=1}^{n} w_i x_i = w_0 + w_1 x_1 + \dots + w_n x_n$

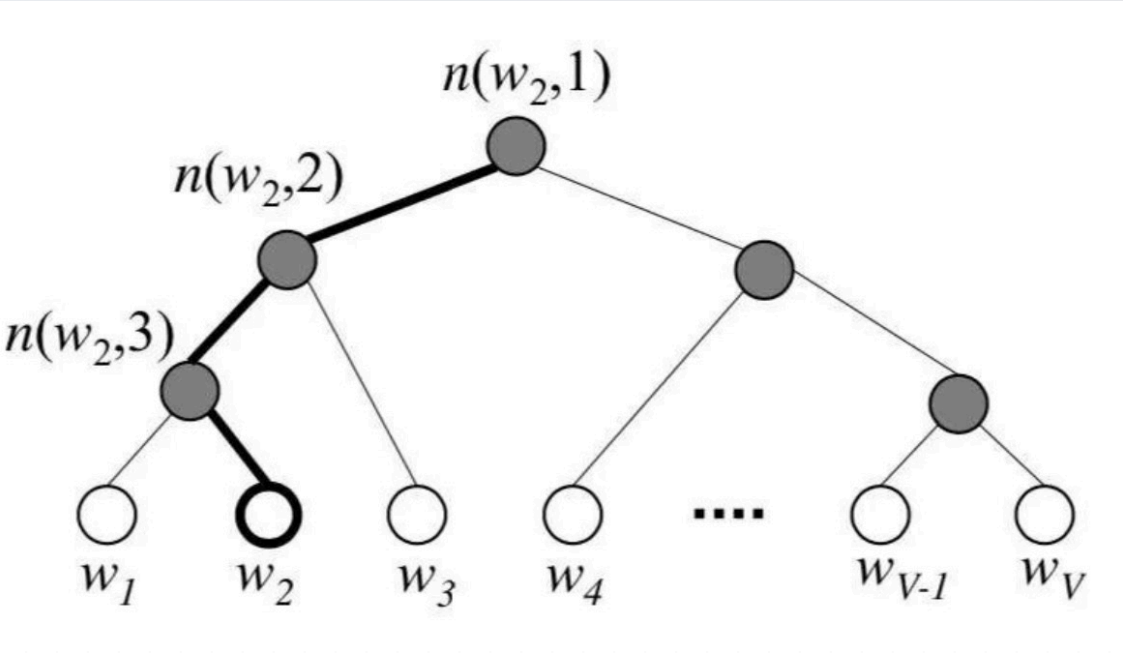
$\sigma(z) = \frac{1}{1+e^{-z}} = \frac{1}{1+e^{-(w^T x+b)}}$

Loss $P(y|x) = \begin{cases} \hat{y} & y=1 \\ 1-\hat{y} & y=0 \end{cases}$

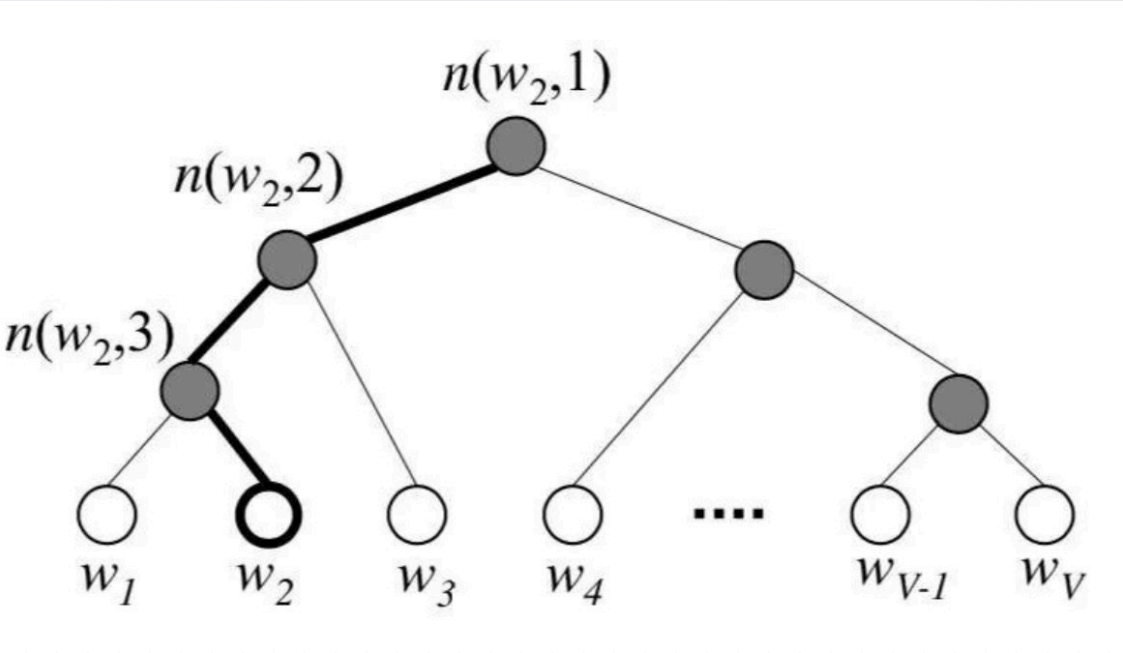
$\log(P(y|x)) = \log(\hat{y}^y (1-\hat{y})^{(1-y)})$

$-(y \log \hat{y} + (1-y) \log(1-\hat{y}))$

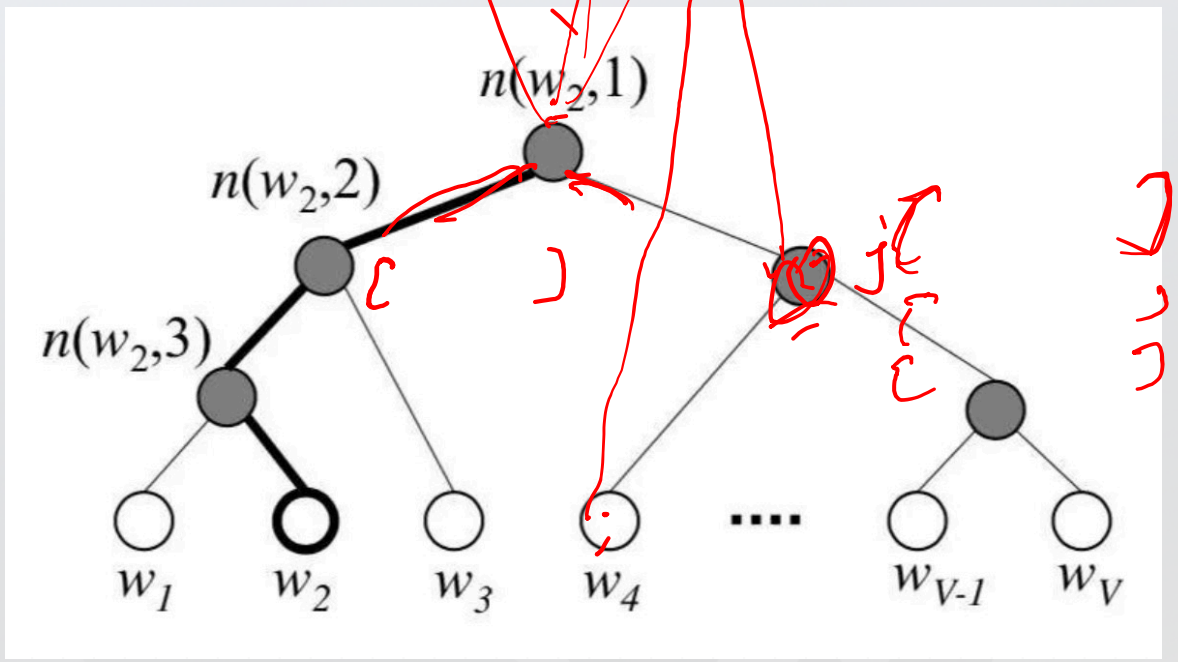
Hierarchical Softmax



Hierarchical Softmax



Hierarchical Softmax



$$p(w|context(w)) = \prod_{j=2}^{l^w} P(d_j^w | x_w, \theta_{j-1}^w)$$

$O(V) \rightarrow O(\log V)$

d_j^w 表示路径第 j 个节点对应编码

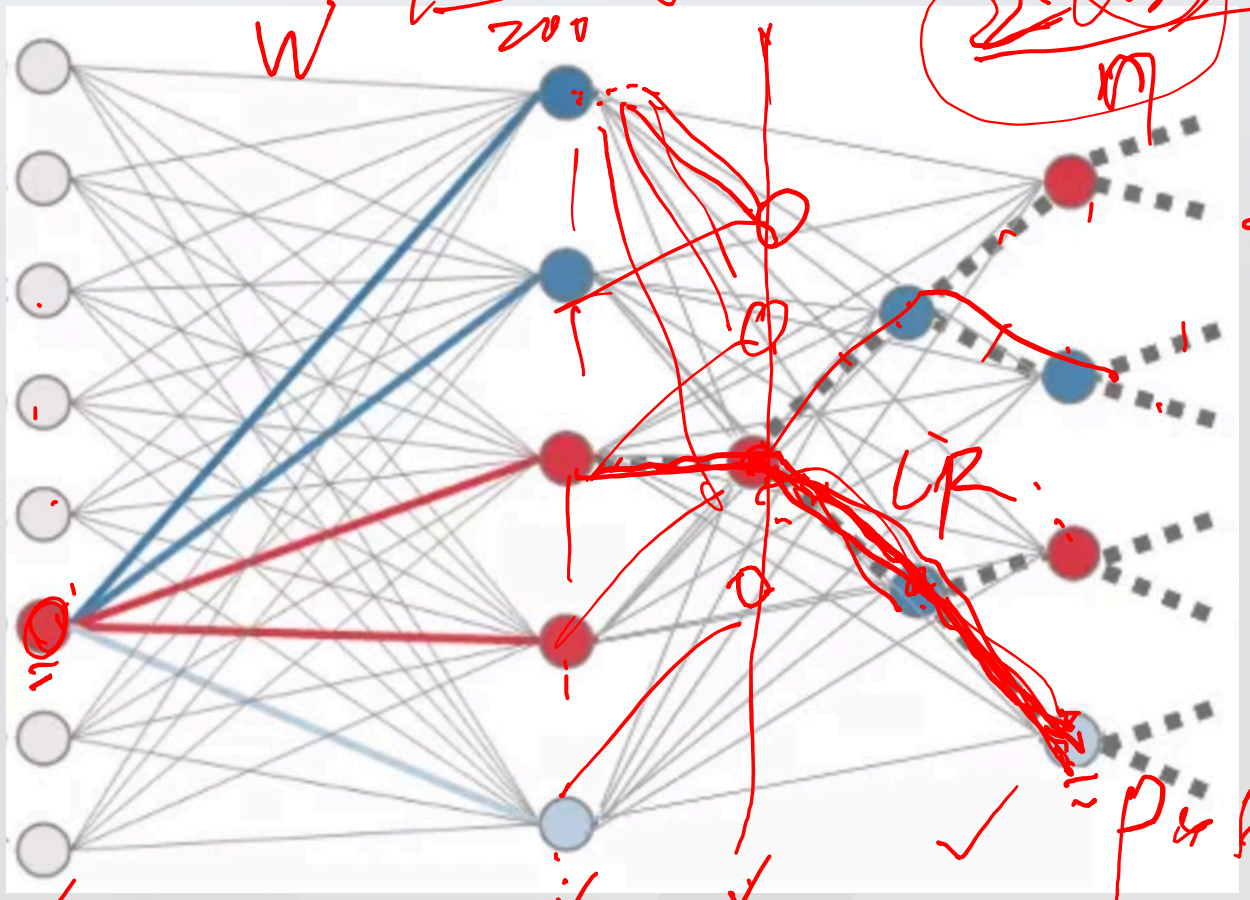
θ_{j-1}^w：路径中非叶子节点对应的参数向量

l^w：路径中包含节点的个数

$$P(d_j^w | x_w, \theta_{j-1}^w) = \begin{cases} \sigma(x_w^T \theta_j^w) & d_j^w = 0 \\ 1 - \sigma(x_w^T \theta_j^w) & d_j^w = 1 \end{cases}$$

$(1-d_j^w) \cdot \log[\ ] + d_j^w \cdot \log[\ ]$

Hierarchical Softmax



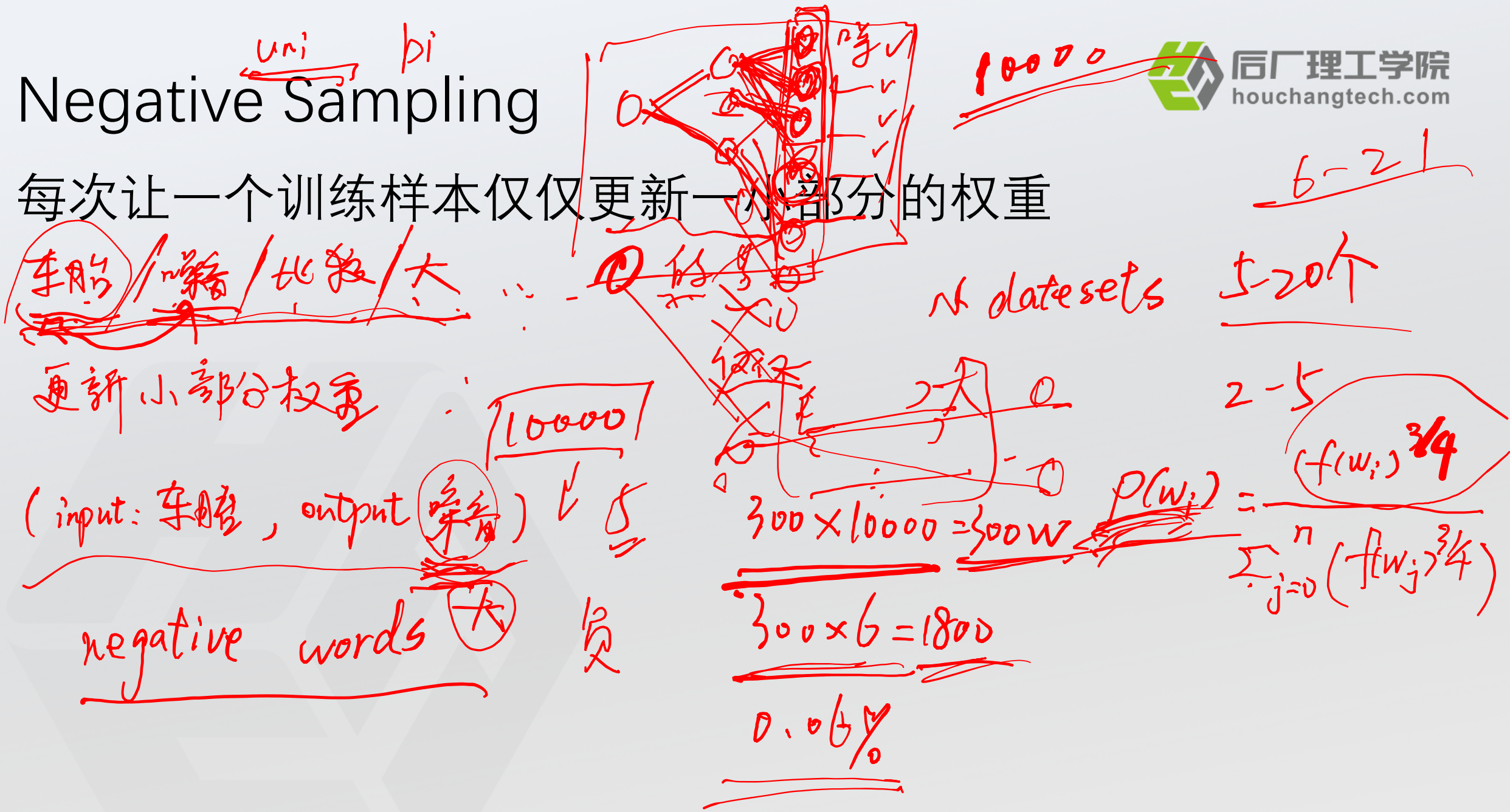
优点：① O(|V|)

→ O(log₂|V|)

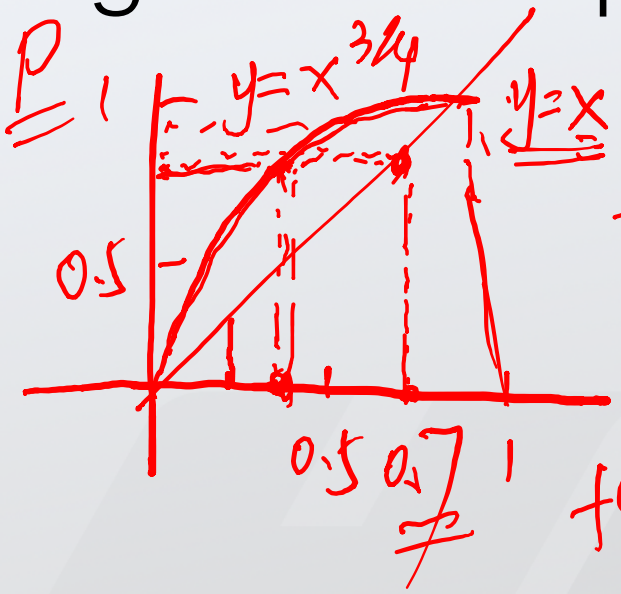
② 高频词路径更短、

Negative Sampling

每次让一个训练样本仅仅更新一小部分的权重



Negative Sampling



(1)

(2) 将长度1线段分为M等份

M >> |V|

(3) len(w) = P(w_i)

从M个未知中采样出Neg个位置

M 10^8

高频词亚采样

2-10倍

w_i $P(w_i) = 1 - \sqrt{\frac{Sample}{Freq(w_i)}}$

$10^{-5} - 10^{-3}$

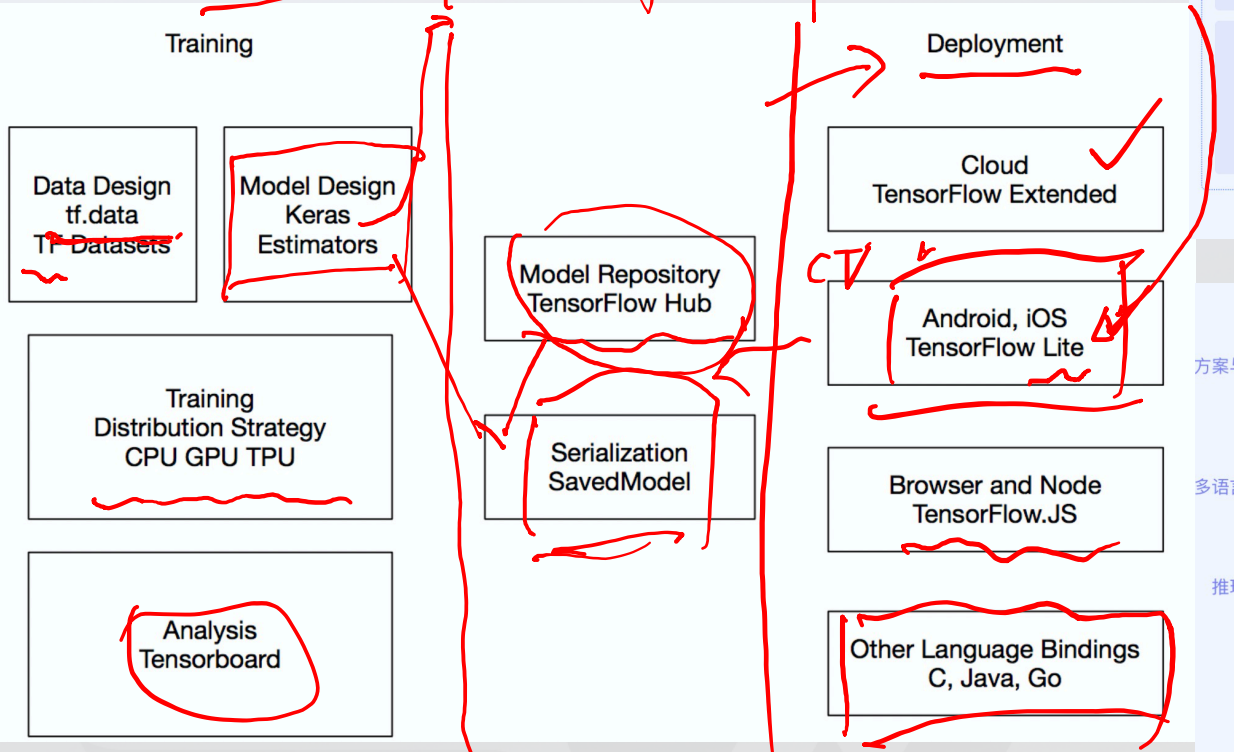
① 对高频词效果更好

② 向量维度较低时效果更好

③ 维度高时近似设定比较大

2. 初识深度学习框架

框架产品结构


Training
Data Design
tf.data
TF Datasets
Model Design
Keras
Estimators
Training
Distribution Strategy
CPU GPU TPU
Analysis
Tensorboard
Model Repository
TensorFlow Hub
Serialization
SavedModel
Deployment
Cloud
TensorFlow Extended
Android, iOS
TensorFlow Lite
Browser and Node
TensorFlow.JS
Other Language Bindings
C, Java, Go


Language Model
Cycle GAN
Se-ResNeXt
Sentiment Classification
OCR
ResNet
NLP
CV
静态图接口
统一Operator接口
动态图接口
组网模块
Executor
图优化
多硬件支持
显存管理
C++模型预测
Kernels
多机分布式
Tracer
Engine
静态图
公用部分
动态图


端到端全流程部署方案
方案与服务
软硬一体方案
部署手册
工具
PaddleSlim
安全与加密
多语言支持
Python
C++
Javascript
推理引擎
Paddle Serving
Paddle Lite
硬件加速库
底层硬件
服务端
移动端
GPU
CPU
新硬件
GPU
CPU
新硬件
NV
AMD
Intel
habanna
寒武纪
BitMain
Metal
Mali
Adreno
ARM
DSP
Cambricon
华为
BitMain

动态图机制

DyGraph

Eager execution

更加灵活便捷的代码组织结构： 使用python的执行控制流程和面向对象的模型设计

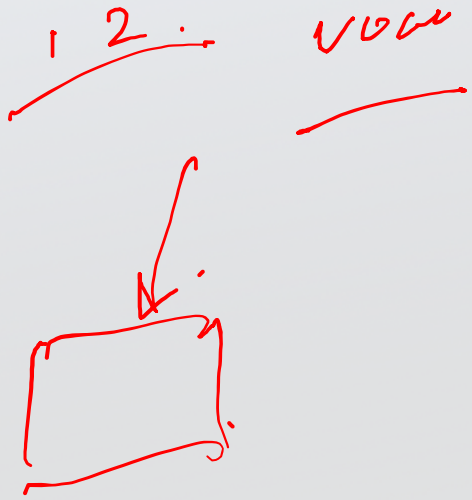
更加便捷的调试功能

```
import paddle.fluid as fluid
import numpy as np

class MyLayer(fluid.dygraph.Layer):
    def __init__(self, name_scope):
        super(MyLayer, self).__init__(name_scope)
        self.fc = fluid.dygraph.nn.FC(self.full_name(), size=12)

    def forward(self, inputs):
        x = self.fc(inputs)
        x = fluid.layers.relu(x)
        self._x_for_debug = x
        x = fluid.layers.elementwise_mul(x, x)
        x = fluid.layers.reduce_sum(x)
        return [x]
```

```
class MyModel(tf.keras.Model):
  def __init__(self):
    super(MyModel, self).__init__()
    self.conv1 = Conv2D(32, 3, activation='relu')
    self.flatten = Flatten()
    self.d1 = Dense(128, activation='relu')
    self.d2 = Dense(10, activation='softmax')

  def call(self, x):
    x = self.conv1(x)
    x = self.flatten(x)
    x = self.d1(x)
    return self.d2(x)
```

Data

对比项	同步Feed方式	异步DataLoader接口方式
API接口	`executor.run(feed=...)`	`fluid.io.DataLoader`
数据格式	Numpy Array或LoDTensor	Numpy Array或LoDTensor
数据增强	Python端使用其他库完成	Python端使用其他库完成
速度	慢	快
推荐用途	调试模型	工业训练



```python
import paddle.fluid as fluid
import numpy as np

# sample级reader
def fake_sample_reader():
    for _ in range(100):
        sample_image = np.random.random(size=(784, )).astype('float32')
        sample_label = np.random.random_integers(size=(1, ), low=0, high=9).astype('int64')
        yield sample_image, sample_label

# batch级reader
def fake_batch_reader():
    batch_size = 32
    for _ in range(100):
        batch_image = np.random.random(size=(batch_size, 784)).astype('float32')
        batch_label = np.random.random_integers(size=(batch_size, 1), low=0, high=9).astype('int64')
        yield batch_image, batch_label
```

Data

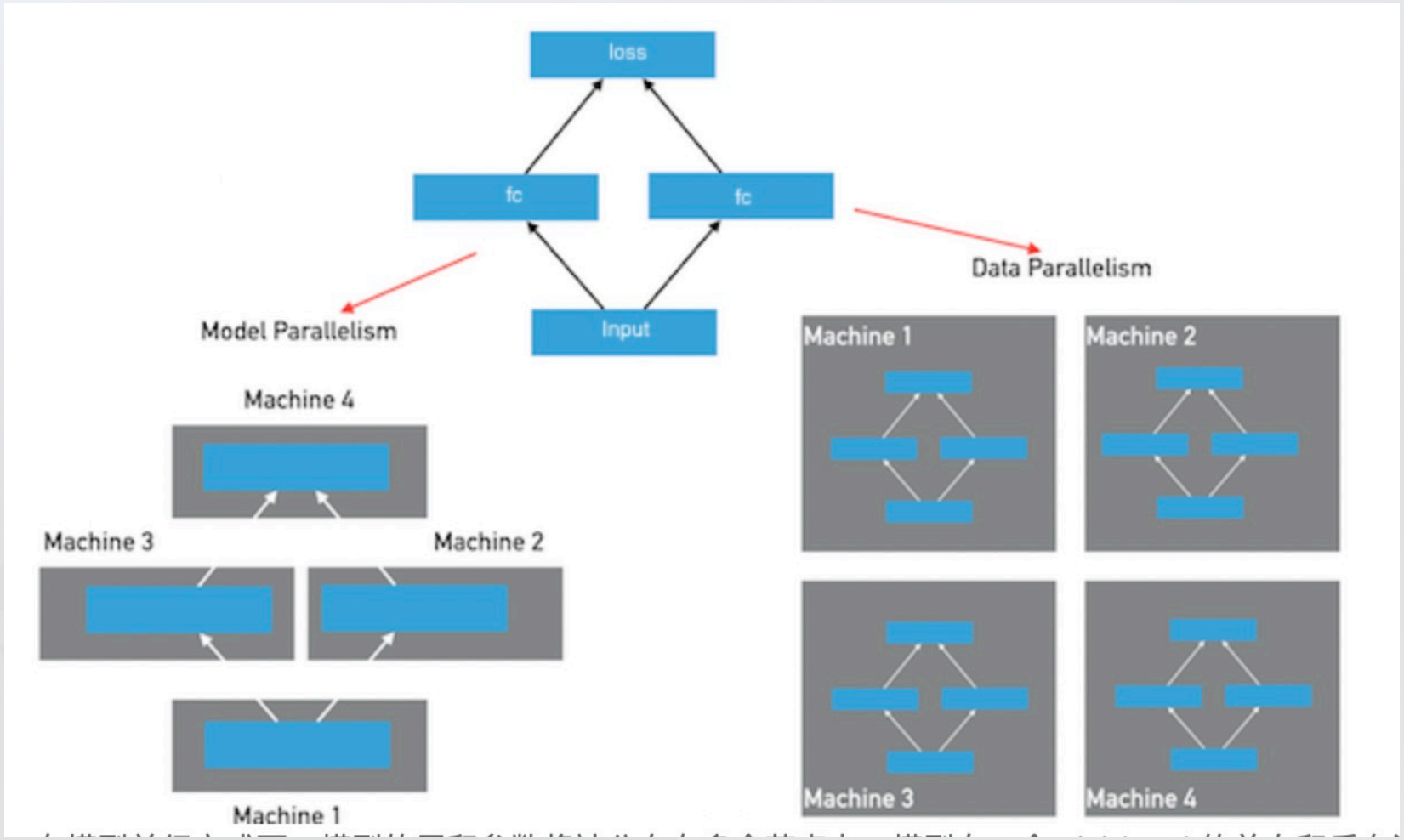

```
import itertools
tf.compat.v1.enable_eager_execution()

def gen():
  for i in itertools.count(1):
    yield (i, [1] * i)

ds = tf.data.Dataset.from_generator(
    gen, (tf.int64, tf.int64), (tf.TensorShape([]), tf.TensorShape([None])))
```

`class TFRecordDataset`: A `Dataset` comprising records from one or more TFRecord files.

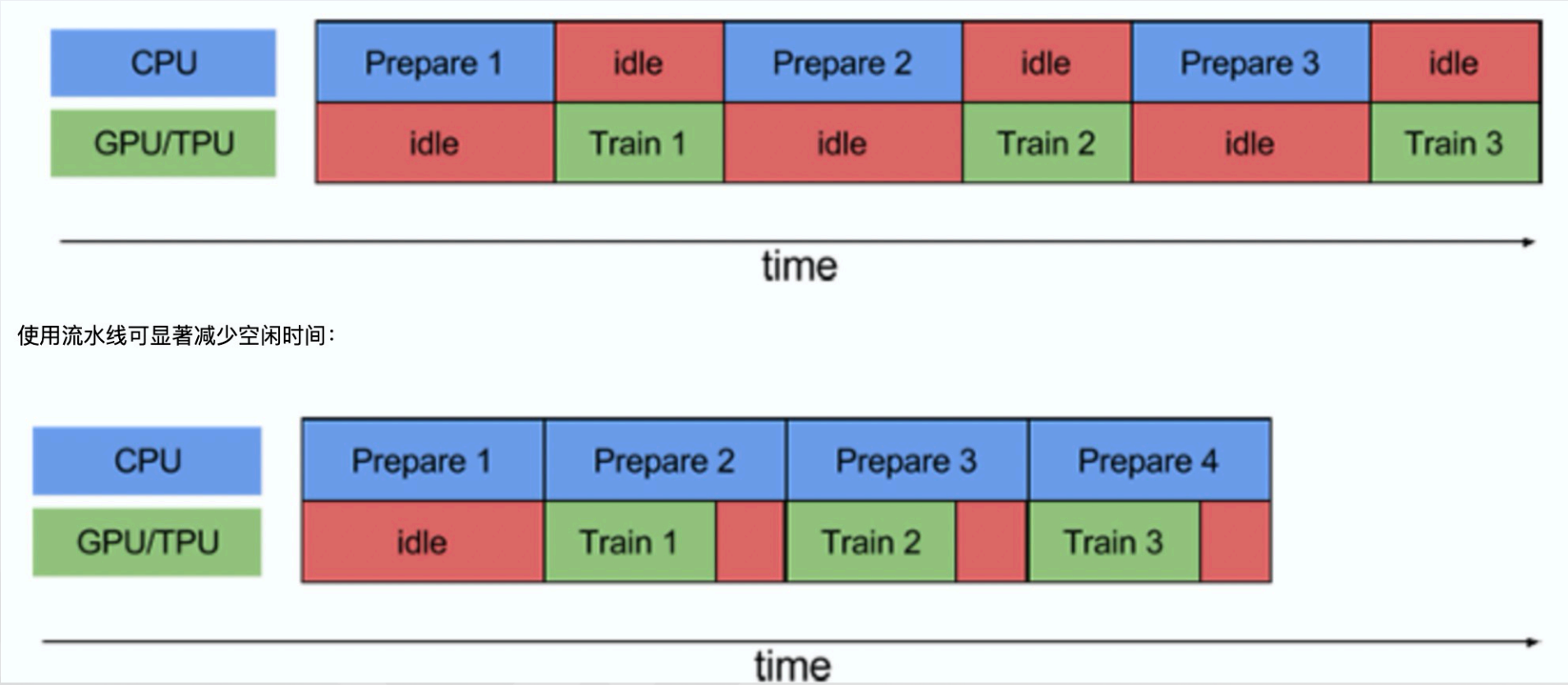
`class TextLineDataset`: A `Dataset` comprising lines from one or more text files.

```
d = tf.data.Dataset.from_tensor_slices([1, 2, 3])
```

分布式训练



分布式训练



使用流水线可显著减少空闲时间：

3. Genism代码实践

10:20

4. 作业

Bye ！