Tensorflow高阶读写教程

原创 2017年04月24日 21:54:19 1851 5 2

前言

tensorflow提供了多种读写方式，我们最常见的就是使用tf.placeholder()这种方法，使用这个方法需要我们提前处理好数据格式，不过这种处理方法也有缺陷：不便于存储和不利于分布式处理，因此，TensorFlow提供了一个标准的读写格式和存储协议，不仅如此，TensorFlow也提供了基于多线程队列的读取方式，高效而简洁，读取速度也更快，据一个博主说速度能提高10倍，相当的诱人.【下面的实验均是在tensorflow1.0的环境下进行】

tensorflow的example解析

example协议

在TensorFlow官方github文档里面，有个example.proto (https://github.com/tensorflow/tensorflow/blob/master/tensorflow/core/example/example.proto)的文件,这个文件详细说明了TensorFlow里面的example协议，下面我将简要叙述一下。

tensorflow的example包含的是基于key-value对的存储方法，其中key是一个字符串，其映射到的是feature信息，feature包含三种类型：

1. BytesList：字符串列表
2. FloatList：浮点数列表
3. Int64List：64位整数列表

以上三种类型都是列表类型，意味着都能够进行拓展,但是也是因为这种弹性格式，所以在解析的时候，需要制定解析参数，这个稍后会讲。

在TensorFlow中，example是**按照行读的**，这个需要时刻记住，比如存储$M \times N$矩阵，使用ByteList存储的话，需要$M \times N$大小的列表，按照每一行的读取方式存放。

tf.tain.example

官方给了一个example的例子：

【转】tensorflow学习使用路线 (/callon…
tensorflow使用tf.dynamic_rnn技巧 (/lyg…
tensorflow dynamic_rnn与static_rnn使…
关于最大似然与交叉熵损失函数和最小…

```
An Example for a movie recommendation application:
  features {
    feature {
      key: "age"
      value { float_list {
        value: 29.0
      }}
    }
    feature {
      key: "movie"
      value { bytes_list {
        value: "The Shawshank Redemption"
        value: "Fight Club"
      }}
    }
    feature {
      key: "movie_ratings"
      value { float_list {
        value: 9.0
        value: 9.7
      }}
    }
    feature {
      key: "suggestion"
      value { bytes_list {
        value: "Inception"
      }}
    }
```

上面的例子中包含一个features，features里面包含一些feature，和之前说的一样，每个feature都是由键值对组成的，其key是一个字符串，其value是上面提到的三种类型之一。

Example中有几个一致性规则需要注意：

1. 如果一个example的feature K 的数据类型是 T，那么所有其他的所有feature K都应该是这个数据类型
2. feature K 的value list的item个数可能在不同的example中是不一样多的，这个取决于你的需求

3. 如果在一个example中没有feature k，那么如果在解析的时候指定一个默认值的话，那么将会返回一个默认值
4. 如果一个feature k 不包含任何的value值，那么将会返回一个空的tensor而不是默认值

tf.train.SequenceExample

sequence_example表示的是一个或者多个sequences，同时还包括上下文context，其中，context表示的是feature_lists的总体特征，如数据集的长度等，feature_list包含一个key，一个value，value表示的是features集合(feature_lists)，同样，官方源码也给出了sequence_example的例子：

```
//ontext: {
  feature: {
    key  : "locale"
    value: {
      bytes_list: {
        value: [ "pt_BR" ]
      }
    }
  }
  feature: {
    key  : "age"
    value: {
      float_list: {
        value: [ 19.0 ]
      }
    }
  }
  feature: {
    key  : "favorites"
    value: {
      bytes_list: {
        value: [ "Majesty Rose", "Savannah Outen", "One Direction" ]
      }
    }
  }
 }
 feature_lists: {
  feature_list: {
    key  : "movie_ratings"
    value: {
      feature: {
        float_list: {
          value: [ 4.5 ]
        }
      }
      feature: {
        float_list: {
```

```
          value: [ 5.0 ]
        }
      }
    }
  }
  feature_list: {
    key  : "movie_names"
    value: {
      feature: {
        bytes_list: {
          value: [ "The Shawshank Redemption" ]
        }
      }
      feature: {
        bytes_list: {
          value: [ "Fight Club" ]
        }
      }
    }
  }
  feature_list: {
    key  : "actors"
    value: {
      feature: {
        bytes_list: {
          value: [ "Tim Robbins", "Morgan Freeman" ]
        }
      }
      feature: {
        bytes_list: {
          value: [ "Brad Pitt", "Edward Norton", "Helena Bonham Carter" ]
        }
      }
    }
  }
}
```

一致性的sequence_example遵循以下规则：

1. context中，所有feature k要保持数据类型一致性
2. 一些example中的某些feature_lists L可能会丢失，如果在解析的时候允许为空的话，那么在解析的时候回返回一个空的list
3. feature_lists可能是空的
4. 如果一个feature_list是非空的，那么其里面的所有feature都**必须是一个数据类型**
5. 如果一个feature_list是非空的，那么对于里面的feature的长度是不是需要一样的，这个取决于解析时候的参数

tensorflow 的parse example解析

在官方代码*[parsing_ops.py](https://github.com/tensorflow/tensorflow/blob/
(https://github.com/tensorflow/tensorflow/blob/)
master/tensorflow/python/ops/parsing_ops.py)*中有关于parse example的详细介绍，我在这里再叙述一下。

tf.parse_example

来看tf.parse_example的方法定义：

```
1 def parse_example(serialized, features, name=None, example_names=None)
```

parse_example是把example解析为词典型的tensor
参数含义：
serialized:一个batch的序列化的example
features:解析example的规则
name：当前操作的名字
example_name:当前解析example的proto名称

这里重点要说的是第二个参数，也就是features，features是把serialized的example中按照键值映射到三种tensor:
1,VarlenFeature 2, SparseFeature 3,FixedLenFeature
下面对这三种映射方式做一个简要的叙述：

VarlenFeature

是按照键值把example的value映射到SpareTensor对象，假设我们有如下的serialized数据：

```
serialized = [
  features
    { feature { key: "ft" value { float_list { value: [1.0, 2.0] } } } },
  features
    { feature []},
  features
    { feature { key: "ft" value { float_list { value: [3.0] } } }
 ]
```

使用VarLenFeatures方法：

```
features={
   "ft":tf.VarLenFeature(tf.float32)
}
```

那么我们将得到的是：

```
{"ft": SparseTensor(indices=[[0, 0], [0, 1], [2, 0]],
                    values=[1.0, 2.0, 3.0],
                    dense_shape=(3, 2)) }
```

可见，显示的indices是ft值的索引，values是值，dense_shape是indices的shape

FixedLenFeature

而FixedLenFeature是按照键值对将features映射到大小为[serilized.size(),df.shape]的矩阵，这里的FixLenFeature指的是每个键值对应的feature的size是一样的。对于上面的例子，如果使用：

```
features: {
    "ft": FixedLenFeature([2], dtype=tf.float32, default_value=-1),
 }
```

那么我们将得到：

```
{"ft": [[1.0, 2.0], [3.0, -1.0]]}
```

可见返回的值是一个[2,2]的矩阵，如果返回的长度不足给定的长度，那么将会使用默认值去填充。

【注意：】

事实上，在TensorFlow1.0环境下，根据官方文档上的内容，我们是能够得到VarLenFeature的值，但是得不到FixLenFeature的值，因此建议如果使用定长的FixedLenFeature，一定要保证对应的数据是等长的。

做个试验来说明：

```
#coding=utf-8

import tensorflow as tf
import os
keys=[[1.0],[],[2.0,3.0]]
sess=tf.InteractiveSession()
sess.run(tf.global_variables_initializer())

def make_example(key):
    example = tf.train.Example(features=tf.train.Features(
        feature={
            'ft':tf.train.Feature(float_list=tf.train.FloatList(value=key))
        }
    ))
    return example

filename="tmp.tfrecords"
if os.path.exists(filename):
    os.remove(filename)
writer = tf.python_io.TFRecordWriter(filename)
for key in keys:
    ex = make_example(key)
    writer.write(ex.SerializeToString())
writer.close()

reader = tf.TFRecordReader()
filename_queue = tf.train.string_input_producer(["tmp.tfrecords"],num_epochs=1)
_,serialized_example =reader.read(filename_queue)

# coord = tf.train.Coordinator()
# threads = tf.train.start_queue_runners(sess=sess,coord=coord)

batch = tf.train.batch(tensors=[serialized_example],batch_size=3)

features={
    "ft":tf.VarLenFeature(tf.float32)
}
```

```
#key_parsed = tf.parse_single_example(make_example([1,2,3]).SerializeToString(),features)
key_parsed = tf.parse_example(batch,features)
#start the queue
print tf.contrib.learn.run_n(key_parsed)

#[]means scalar

features={
   "ft":tf.FixedLenFeature(shape=[2],dtype=tf.float32)
}

key_parsed = tf.parse_example(batch,features)

print tf.contrib.learn.run_n(key_parsed)

```

结果返回如下：

```
[{'ft': SparseTensorValue(indices=array([[0, 0],
       [2, 0],
       [2, 1]]), values=array([ 1.,  2.,  3.], dtype=float32), dense_shape=array([3, 2]))}]

InvalidArgumentError (see above for traceback): Name: <unknown>, Key: ft, Index: 0.  Number of float values
```

可见，对于VarLenFeature，是能返回正常结果的，但是对于FixedLenFeature则返回size不对，可见如果对于边长的数据还是不要使用FixedLenFeature为好。

如果把数据设置为[[1.0,2.0],[2.0,3.0]],那么FixedLenFeature返回的是：

```
[{'ft': array([[ 1.,  2.],
       [ 2.,  3.]], dtype=float32)}]
```

这是正确的结果。

SparseFeature可以从下面的例子来说明：

```
`serialized`:
 ```
 [
  features {
    feature { key: "val" value { float_list { value: [0.5, -1.0] } } }
    feature { key: "ix" value { int64_list { value: [3, 20] } } }
  },
  features {
    feature { key: "val" value { float_list { value: [0.0] } } }
    feature { key: "ix" value { int64_list { value: [42] } } }
  }
 ]
 ```
 And arguments
 ```
 example_names: ["input0", "input1"],
 features: {
    "sparse": SparseFeature(
       index_key="ix", value_key="val", dtype=tf.float32, size=100),
 }
 ```
 Then the output is a dictionary:
 ```python
 {
  "sparse": SparseTensor(
      indices=[[0, 3], [0, 20], [1, 42]],
      values=[0.5, -1.0, 0.0]
      dense_shape=[2, 100]),
 }
 ```
```

现在明白了Example的协议和tf.parse_example的方法之后，我们再看看看几个简单的parse_example

tf.parse_single_example

区别于tf.parse_example，tf.parse_single_example只是少了一个batch而已，其余的都是一样的，我们看代码：

```
#coding=utf-8

import tensorflow as tf
import os

sess=tf.InteractiveSession()
sess.run(tf.global_variables_initializer())

def make_example(key):
    example = tf.train.Example(features=tf.train.Features(
        feature={
            'ft':tf.train.Feature(float_list=tf.train.FloatList(value=key))
        }
    ))
    return example

features={
    "ft":tf.FixedLenFeature(shape=[3],dtype=tf.float32)
}

key_parsed = tf.parse_single_example(make_example([1.0,2.0,3.0]).SerializeToString(),features)

print tf.contrib.learn.run_n(key_parsed)

```

结果返回为：

```
[{'ft': array([ 1.,  2.,  3.], dtype=float32)}]
```

tf.parse_single_sequence_example

tf.parse_single_sequence_example对应的是tf.train,SequenceExample,我们以下面代码说明，single_sequence_example的用法：

```
#coding=utf-8

import tensorflow as tf
import os
keys=[[1.0,2.0],[2.0,3.0]]
sess=tf.InteractiveSession()
sess.run(tf.global_variables_initializer())

def make_example(locale,age,score,times):

    example = tf.train.SequenceExample(
        context=tf.train.Features(
            feature={
            "locale":tf.train.Feature(bytes_list=tf.train.BytesList(value=[locale])),
            "age":tf.train.Feature(int64_list=tf.train.Int64List(value=[age]))
        }),
        feature_lists=tf.train.FeatureLists(
            feature_list={
            "movie_rating":tf.train.FeatureList(feature=[tf.train.Feature(float_list=tf.train.FloatList(value=score)) for
            }
        )
    )
    return example.SerializeToString()

context_features = {
    "locale": tf.FixedLenFeature([],dtype=tf.string),
    "age": tf.FixedLenFeature([],dtype=tf.int64)
}
sequence_features = {
    "movie_rating": tf.FixedLenSequenceFeature([3], dtype=tf.float32,allow_missing=True)
}

context_parsed, sequence_parsed  = tf.parse_single_sequence_example(make_example("china",24,[1.0,3.5,4.0

print tf.contrib.learn.run_n(context_parsed)
print tf.contrib.learn.run_n(sequence_parsed)
```

结果打印为：

```
[{'locale': 'china', 'age': 24}]

[{'movie_rating': array([[ 1. ,  3.5,  4. ],
       [ 1. ,  3.5,  4. ]], dtype=float32)}]
```

tf.parse_single_sequence_example的自动补齐

在常用的文本处理方面，由于文本经常是非定长的，因此需要经常补齐操作，例如使用CNN进行文本分类的时候就需要进行padding操作，通常我们把padding的索引设置为0，而且在文本预处理的时候也需要额外的代码进行处理，而TensorFlow提供了一个比较好的自动补齐工具，就是在tf.train.batch里面把参数dynamic_pad设置成True，样例如下：

```
#coding=utf-8

import tensorflow as tf
import os
keys=[[1,2],[2]]
sess=tf.InteractiveSession()
sess.run(tf.global_variables_initializer())



def make_example(key):

    example = tf.train.SequenceExample(
        context=tf.train.Features(
            feature={
            "length":tf.train.Feature(int64_list=tf.train.Int64List(value=[len(key)]))
        }),
        feature_lists=tf.train.FeatureLists(
            feature_list={
            "index":tf.train.FeatureList(feature=[tf.train.Feature(int64_list=tf.train.Int64List(value=[key[i]])) for i in r
            }
        )
    )
    return example.SerializeToString()


filename="tmp.tfrecords"
if os.path.exists(filename):
    os.remove(filename)
writer = tf.python_io.TFRecordWriter(filename)
for key in keys:
    ex = make_example(key)
    writer.write(ex)
writer.close()

reader = tf.TFRecordReader()
filename_queue = tf.train.string_input_producer(["tmp.tfrecords"],num_epochs=1)
```

```
_,serialized_example =reader.read(filename_queue)

# coord = tf.train.Coordinator()
# threads = tf.train.start_queue_runners(sess=sess,coord=coord)

context_features={
    "length":tf.FixedLenFeature([],dtype=tf.int64)
}
sequence_features={
    "index":tf.FixedLenSequenceFeature([],dtype=tf.int64)
}

context_parsed, sequence_parsed = tf.parse_single_sequence_example(
    serialized=serialized_example,
    context_features=context_features,
    sequence_features=sequence_features
)

batch_data = tf.train.batch(tensors=[sequence_parsed['index']],batch_size=2,dynamic_pad=True)
result = tf.contrib.learn.run_n({"index":batch_data})

print result
```


打印结果如下：

```
[{'index': array([[1, 2],
       [2, 0]])}]
```

可见还是比较好用的功能

tensorflow的TFRecords读取

在上面的部分，我们展示了关于tensorflow的example的用法和解析过程，那么我们该如何使用它们呢？其实在上面的几段代码里面也有体现，就是TFRecords进行读写，TFRecords读写其实很简单,tensorflow提供了两个方法：

1. tf.TFRecordReader
2. tf.TFRecordWriter

首先我们看下第二个，也就是tf.TFRecordWritre，之所以先看第二个的原因是第一个Reader将和batch一起在下一节讲述。

关于TFRecordWriter,可以用下面代码说明,假设serilized_object是一个已经序列化好的example，那么其写的过程如下：

```
writer = tf.python_io.TFRecordWriter(filename)
writer.write(serilized_object)
writer.close()
```

tensorflow的多线程batch读取

这一节主要关注的是基于TFRecords的读取的方法和batch操作，我们可以回看一下之前的博客(http://blog.csdn.net/u010223750/article/details/53244744)的batch操作：

```
Batching

def read_my_file_format(filename_queue):
  reader = tf.SomeReader()
  key, record_string = reader.read(filename_queue)
  example, label = tf.some_decoder(record_string)
  processed_example = some_processing(example)
  return processed_example, label

def input_pipeline(filenames, batch_size, num_epochs=None):
  filename_queue = tf.train.string_input_producer(
      filenames, num_epochs=num_epochs, shuffle=True)
  example, label = read_my_file_format(filename_queue)
  # min_after_dequeue defines how big a buffer we will randomly sample
  #   from -- bigger means better shuffling but slower start up and more
  #   memory used.
  # capacity must be larger than min_after_dequeue and the amount larger
  #   determines the maximum we will prefetch.  Recommendation:
  #   min_after_dequeue + (num_threads + a small safety margin) * batch_size
  min_after_dequeue = 10000
  capacity = min_after_dequeue + 3 * batch_size
  example_batch, label_batch = tf.train.shuffle_batch(
      [example, label], batch_size=batch_size, capacity=capacity,
      min_after_dequeue=min_after_dequeue)
  return example_batch, label_batch
```

这里我们把tf.SomeReader()换成tf.TFRecordReader()即可，然后再把tf.some_decoder换成我们自定义的decoder，当然在decoder里面我们可以自己指定parser（也就是上文提到的内容），然后我们使用tf.train.batch或者tf.train.shuffle_batch等操作获取到我们需要送入网络训练的batch参数即可。

多线程读取batch实例

我使用了softmax回归做一个简单的示例，下面是一个多线程读取batch的实例主要代码：

```
#coding=utf-8
"""
author:luchi
date:24/4/2017
desc:training logistic regression
"""
import tensorflow as tf
from model import Logistic

def read_my_file_format(filename_queue):
    reader = tf.TFRecordReader()
    _,serilized_example = reader.read(filename_queue)

    #parsing example
    features = tf.parse_single_example(serilized_example,
        features={
            "data":tf.FixedLenFeature([2],tf.float32),
            "label":tf.FixedLenFeature([],tf.int64)
        }

    )

    #decode from raw data,there indeed do not to change ,but to show common step , i write a case here

    # data = tf.cast(features['data'],tf.float32)
    # label = tf.cast(features['label'],tf.int64)

    return features['data'],features['label']


def input_pipeline(filenames, batch_size, num_epochs=100):


    filename_queue = tf.train.string_input_producer([filenames],num_epochs=num_epochs)
    data,label=read_my_file_format(filename_queue)

    datas,labels = tf.train.shuffle_batch([data,label],batch_size=batch_size,num_threads=5,
```

```
                                capacity=1000+3*batch_size,min_after_dequeue=1000)
    return datas,labels

class config():
    data_dim=2
    label_num=2
    learining_rate=0.1
    init_scale=0.01

def run_training():

    with tf.Graph().as_default(), tf.Session() as sess:

        datas,labels = input_pipeline("reg.tfrecords",32)

        c = config()
        initializer = tf.random_uniform_initializer(-1*c.init_scale,1*c.init_scale)

        with tf.variable_scope("model",initializer=initializer):
            model = Logistic(config=c,data=datas,label=labels)

        fetches = [model.train_op,model.accuracy,model.loss]
        feed_dict={}

        #init
        init_op = tf.group(tf.global_variables_initializer(),
                    tf.local_variables_initializer())
        sess.run(init_op)

        coord = tf.train.Coordinator()
        threads = tf.train.start_queue_runners(sess=sess,coord=coord)
        try:
            while not coord.should_stop():

                # fetches = [model.train_op,model.accuracy,model.loss]
                # feed_dict={}
                # feed_dict[model.data]=sess.run(datas)
```

```
            # feed_dict[model.label]=sess.run(labels)
            # _,accuracy,loss= sess.run(fetches,feed_dict)
            _,accuracy,loss= sess.run(fetches,feed_dict)
            print("the loss is %f and the accuracy is %f"%(loss,accuracy))
        except tf.errors.OutOfRangeError:
            print("done training")
        finally:
            coord.request_stop()
        coord.join(threads)
        sess.close()

def main():
    run_training()

if __name__=='__main__':
    main()
```

这里有几个坑需要说明一下：

1. 使用了string_input_producer指定num_epochs之后，在初始化的时候需要使用：

```
init_op = tf.group(tf.global_variables_initializer(),
                   tf.local_variables_initializer())
sess.run(init_op)
```

要不然会报错

2. **使用了从文件读取batch之后，就不需要设置tf.placeholder了【非常重要】，**我在这个坑里呆了好久，如果使用了tf.placeholder一是会报错为tensor对象能送入到tf.placeholder中，另外一个是就算使用sess.run(batch_data),也会存在模型不能收敛的问题，所以切记切记

结果显示如下：

```
the loss is 0.156685 and the accuracy is 0.937500
the loss is 0.185438 and the accuracy is 0.968750
the loss is 0.092628 and the accuracy is 0.968750
the loss is 0.059271 and the accuracy is 1.000000
the loss is 0.088685 and the accuracy is 0.968750
the loss is 0.271341 and the accuracy is 0.968750
the loss is 0.244190 and the accuracy is 0.968750
the loss is 0.136841 and the accuracy is 0.968750
the loss is 0.115607 and the accuracy is 0.937500
the loss is 0.080254 and the accuracy is 1.000000
```

目录

完整的代码见我的GitHub (https://github.com/luchi007/TFrecordsBatch)

喜欢

后记

收藏

关于tfrecords的batch操作，我一共折腾了好几天，中间也是磕磕绊绊，走过各种坑，现在也终于算是明白了一些，也算是没白费功夫，不过使用这种特性之后，一是提高了速度，而是减少了写代码的量，当然因为数据成了序列化的，也容易检查数据是否有误，最后，希望明天面试顺利吧，2017年4月24，于北京

评论

分享

标签：tensorflow (http://so.csdn.net/so/search/s.do?q=tensorflow&t=blog) / TFRecords (http://so.csdn.net/so/search/s.do?q=TFRecords&t=blog) /

返回顶部

qq_38184501 (/qq_38184501) 2017-05-05 10:27 4

SEQUENCES,LABELS,LENGTH=READ_MYFILE_FORMAT(FILENAME_QUEUE)
FOR I IN RANGE(4):
BATCH_SEQUENCES,BATCH_LABELS=TF.TRAIN.BATCH([SEQUENCES,LABELS],BATCH_SIZE=2,DYNAMIC_PAD=TRU
RESULT=TF.CONTRIB.LEARN.RUN_N({'SEQUENCES':BATCH_SEQUENCES,'LABELS':BATCH_LABELS})
PRINT RESULT
我想这样显示出来三个BATCH的数据，结果发现显示出来的后两个BATCH的数据完全是乱序的

我想这样显示出来三个batch的数据，结果发现显示出来的后两个batch的数据完全是乱序的

回复

qq_38184501 (/qq_38184501) 2017-05-03 20:55 3楼

博主你好！我最近在做序列分类问题，要用到lstm。
还想问一个问题：对于我写入的序列，就是用SequenceExample写好的TFRcords，也可以用最后所说的方法进行batch读取吗？有什么区别吗？

回复 1条回复

查看 5 条热评

相关文章推荐

tensorflow中cifar-10文档的Read操作 (/u010223750/article/details/53244744)

前言 在tensorflow的官方文档中得卷积神经网络一章，有一个使用cifar-10图片数据集的实验，搭建卷积神经网络倒不难，但是那个cifar10_input文件着实让我费了一番心思。配合着官方...

u010223750 (http://blog.csdn.net/u010223750) 2016-11-20 22:29 1586

【转】tensorflow学习使用路线 (/callon_h/article/details/60343633)

版权声明：本文为博主hjimce的原创文章，原文地址为http://blog.csdn.net/hjimce/article/details/51899683。

Callon_H (http://blog.csdn.net/Callon_H) 2017-03-04 20:37 968

精选：深入理解 Docker 内部原理及网络配置 (http://edu.csdn.net/huiyiCourse/detail/563?utm_source=blog10)

网络绝对是任何系统的核心，对于容器而言也是如此。Docker 作为目前最火的轻量级容器技术，有很多令人称道的功能，如 Docker 的镜像管理。然而，Docker的网络一直以来都比较薄弱，所以我们有必要深入了解Docker的网络知识，以满足更高的网络需求。

tensorflow使用tf.dynamic_rnn技巧 (/lyg5623/article/details/73924506)

用rnn处理变长文本时，使用dynamic_rnn可以跳过padding部分的计算，减少计算量。假设有两个文本，一个长度为10，另一个长度为5，那么需要对第二文本使用0-padding方法填充，得到的...

lyg5623 (http://blog.csdn.net/lyg5623) 2017-06-29 18:37 650

tensorflow dynamic_rnn与static_rnn使用注意 (/daxiaofan/article/details/70197812)

不同有好多，例如：输入输出输入输出要格外注意，敲代码的时候，这个和我们关系最大 帖俩段代码，注意其中的输入输出这个是static_rnndef lstm_model(x,y): # x = tf...

daxiaofan (http://blog.csdn.net/daxiaofan) 2017-04-16 20:33 2432

关于最大似然与交叉熵损失函数和最小二乘法的思考 (/u010223750/article/details/52747895)

最大似然估计与logistic交叉熵损失函数以及线性回归过程中的最小二乘法的关系理解

u010223750 (http://blog.csdn.net/u010223750) 2016-10-07 15:07 1962

tensorflow高阶教程:tf.dynamic_rnn (/u010223750/article/details/71079036)

喜欢|言TensorFlow很容易上手，但是TensorFlow的很多trick却是提升TensorFlow心法的法门，之前说过TensorFlow的read心法，现在想说一说TensorFlow在RNN...

u010223750 (http://blog.csdn.net/u010223750) 2017-05-02 11:27 4723

深度学习（08）_RNN-LSTM循环神经网络-03-Tensorflow进阶实现 (/u013082989/article/details/73693392)

全部代码：点击这里查看 本文个人博客地址：点击这里查看 关于Tensorflow实现一个简单的二元序列的例子可以点击这里查看 关于RNN和LSTM的基础可以查看这里 这篇博客主要包含以下内容 训练一...

u013082989 (http://blog.csdn.net/u013082989) 2017-06-24 18:29 2337

tensorflow中的RNN与LSTM函数异同点分析 (/liuweizj12/article/details/77862333)

import tensorflow as tf import numpy as np # X = np.random.randn(2, 5, 2) X = np.array([[[0.1,0.2],...

liuweizj12 (http://blog.csdn.net/liuweizj12) 2017-09-06 09:39 24

目录

喜欢

收藏

评论

分享

返回顶部

TensorFlow高效读取数据的方法 (/u012759136/article/details/52232266)

概述关于Tensorflow读取数据，官网给出了三种方法： 供给数据(Feeding)： 在TensorFlow程序运行的每一步， 让Python代码来供给数据。 从文件读取数据： 在TensorFl...

 u012759136 (http://blog.csdn.net/u012759136) 2016-08-17 19:20 25121

Tensorflow中使用tfrecord方式读取数据 (/u010358677/article/details/70544241)

前言本博客默认读者对神经网络与Tensorflow有一定了解，对其中的一些术语不再做具体解释。并且本博客主要以图片数据为例进行介绍，如有错误，敬请斧正。使用Tensorflow训练神经网络时，我们可以...

 u010358677 (http://blog.csdn.net/u010358677) 2017-04-23 20:11 4377

Tensorflow高阶读写教程 (/u010223750/article/details/70482498)

前言tensorflow提供了多种读写方式，我们最常见的就是使用tf.placeholder()这种方法，使用这个方法需要我们提前处理好数据格式，不过这种处理方法也有缺陷：不便于存储和不利于分布式处理...

 u010223750 (http://blog.csdn.net/u010223750) 2017-04-24 21:54 1852

Android 中LayoutInflater的使用! (/heimabb/article/details/8438800)

大家好我们这一节讲的是LayoutInflater的使用,在实际开发种LayoutInflater这个类还是非常有用的,它的作用类似于 findViewById(), 不同点是LayoutInfla...

 heimabb (http://blog.csdn.net/heimabb) 2012-12-26 14:04 146

Tensorflow: 文件读写 (/qq_39037910/article/details/72900308)

写文件这里的写, 指的是把各种格式的数据(如字符, 图片等)统一转换成Tensorflow的标准支持格式TFRecord.TFRecord是输入数据统一管理的格式, 它其实是一种二进制文件. 写入...

qq_39037910 (http://blog.csdn.net/qq_39037910) 2017-06-07 15:31 162

Mycat高级进阶---读写分离 (/wangshuang1631/article/details/64904871)

MySQL主从复制的几种方案数据库读写分离对于大型系统或者访问量很高的互联网应用来说，是必不可少的一个重要功能。从数据库的角度来说，对于大多数应用来说，从集中到分布，最基本的一个需求不是数据存储的瓶...

wangshuang1631 (http://blog.csdn.net/wangshuang1631) 2017-03-22 09:36 1037

Android高手进阶教程(二十三)之---Android中的日历读写操作!!! (/liangxiaozhang/article/details/7496847)

大家好,好久没有更新blog了，今天给大家分享一下Android中一些自带日历的操作方法,这里主要用到了ContentProiver的知识.如果大家不明白ContentProvider建议先查一下资料...

liangxiaozhang (http://blog.csdn.net/liangxiaozhang) 2012-04-25 10:39 529

Android高手进阶教程(二十三)之---Android中的日历读写操作!!! (/android_tutor/article/details/6165470)

大家好,好久没有更新blog了，今天给大家分享一下Android中一些自带日历的操作方法,这里主要用到了ContentProiver的知识.如果大家不明白ContentProvider建议先查一下资料...

Android_Tutor (http://blog.csdn.net/Android_Tutor) 2011-01-26 21:20 35487

Android高手进阶教程(二十三)之---Android中的日历读写操作!!! (/javatiger427/article/details/6594250)

大家好,好久没有更新blog了，今天给大家分享一下Android中一些自带日历的操作方法,这里主要用到了ContentProiver的知识.如果大家不明白ContentProvider建议先查一下资料...

 JavaTiger427 (http://blog.csdn.net/JavaTiger427) 2011-07-09 10:11 1176

Android高手进阶教程(二十)---Android中的日历读写操作!!! (/peter_hucq/article/details/6777224)

大家好,好久没有更新blog了，今天给大家分享一下Android中一些自带日历的操作方法,这里主要用到了ContentProiver的知识.如果大家不明白ContentProvider建议先查一下资料...

 Peter_hucq (http://blog.csdn.net/Peter_hucq) 2011-09-15 10:28 1103

[TensorFlow学习笔记1]TensorFLow的基本概念和基本使用 (/qq_17506541/article/details/68942412)

这里先给出参考链接： https://github.com/jikexueyuanwiki/tensorflow-zh/blob/master/SOURCE/get_started/basic_us...

 qq_17506541 (http://blog.csdn.net/qq_17506541) 2017-04-01 11:12 59

Tensorflow学习Lecture 1 (/qq_26609915/article/details/52444883)

TensorFlow 是一个编程系统, 使用图来表示计算任务. 图中的节点被称之为op (operation 的缩写). 一个 op获得 0 个或多个Tensor , 执行计算, 产生 0 个或多个T...

 qq_26609915 (http://blog.csdn.net/qq_26609915) 2016-09-05 23:17 113

目录

喜欢

收藏

评论

分享

返回顶部