TensorFlow 2.x NLP Best Practice

QHDuan 2020-11-14 DevFest 2020

TextVectorization

TextVectorization：问题

1. 词表处理程序和模型分离
2. 部署时的处理与效率

TextVectorization：方案

```
x = [ '你 好', 'I love you' ]
# 构建层
text_vector =
tf.keras.layers.experimental.preprocessing.TextVectorization()
# 学习词表
text_vector.adapt(x)
print(text_vector(x))
# 输出：
# tf.Tensor(
# [[4 2 0]
#  [7 6 5]], shape=(2, 3), dtype=int64)
```

TextVectorization：模型

```
tf.keras.Sequential([
    text_vector,
    tf.keras.layers.Embedding(
        len(text_vector.get_vocabulary()),
        32
    ),
    tf.keras.layers.Dense(2)
])
```

TextVectorization如何实现？

tf.strings

tf.strings：专门处理字符串类型

format
join
length
split
strip
substr
lower
upper

ngrams
regex_full_match
regex_replace
to_hash_bucket
unicode_decode
unicode_encode
unicode_split
bytes_split
To_number
as_string
reduce_join

tf.strings：切割字符串

```
x = [
    '你 好 啊',
    'I love you'
]
print(tf.strings.split(x))
# 输出：
# <tf.RaggedTensor [
#    [ b'\xe4\xbd\xa0', b'\xe5\xa5\xbd', b'\xe5\x95\x8a'],
#    [b'I', b'love', b'you']]>
```

tf.strings：tf.lookup.StaticHashTable实现词表

```
keys_tensor = tf.constant(['你', '好', '啊'])
vals_tensor = tf.constant([1, 2, 3])
table = tf.lookup.StaticHashTable(
    tf.lookup.KeyValueTensorInitializer(keys_tensor, vals_tensor), -1)
print(table.lookup(tf.constant(['你', '好'])))
# 输出：
# tf.Tensor([1 2], shape=(2,), dtype=int32)
```

bucket_by_sequence_length

Bucket_by_sequence_length：问题

NLP的训练效率，它可能跟其他任务有什么区别？

NLP主要处理不定长数据

Bucket_by_sequence_length：现象

Bucket_by_sequence_length：方案

```
def data_generator():
    while True:
        for length in (1, 9, 2, 8, 3, 7, 4, 6):
            yield 'I' * length

dataset = tf.data.Dataset.from_generator(
    data_generator, output_types=tf.string,
    output_shapes=tf.TensorShape([]))
```

Bucket_by_sequence_length：方案

```
for x in dataset.take(3):
    print(x)

# tf.Tensor(b'I', shape=(), dtype=string)
# tf.Tensor(b'IIIIIIII', shape=(), dtype=string)
# tf.Tensor(b'II', shape=(), dtype=string)
```

Bucket_by_sequence_length：方案

```
# dataset = ...
bucket_boundaries = [4, 8]
bucket_batch_sizes = [4, 3, 2]
dataset = dataset.apply(

tf.data.experimental.bucket_by_sequence_length(
        element_length_func=tf.strings.length,
        bucket_batch_sizes=bucket_batch_sizes,
        bucket_boundaries=bucket_boundaries
    )
)
```

Bucket_by_sequence_length：方案

```
for x in dataset.take(3):
    print(x)

# tf.Tensor([b'IIIIIIII' b'IIIIIII'], shape=(2,), dtype=string)
# tf.Tensor([b'IIIIII' b'III' b'IIIII'], shape=(3,), dtype=string)
# tf.Tensor([b'I' b'II' b'III' b'I'], shape=(4,), dtype=string)
```

TensorFlow Text

TensorFlow Text : BERT

```
import tensorflow as tf
import tensorflow_text as text

t = text.BertTokenizer(
    './pretrained/chinese_L-12_H-768_A-12/vocab.txt')

r = t.tokenize(['你好', '嗷嗷'])

print(r)
# <tf.RaggedTensor [[[872], [1962]], [[1643], [1643]]]>
```

TensorFlow Text : BERT

```
print(r.to_tensor())
# tf.Tensor(
# [[[ 872]
#   [1962]]
#  [[1643]
#   [1643]]], shape=(2, 2, 1), dtype=int64)
```

THANK YOU

QHDuan 2020-11-14 DevFest 2020