TensorFlow Everywhere
Korea

정욱재
머신러닝 엔지니어 - 당근마켓

편리한 NLP를 위한 TensorFlow-Text와 RaggedTensor

정욱재

당근마켓에서 머신러닝 엔지니어로 일하고 있어요.

경량화/NLP 분야의 머신러닝과 더불어
TensorFlow, PyTorch 등 딥러닝 라이브러리 “자체”에 관심이 많아요.

- https://github.com/jeongukjae
- https://jeongukjae.github.io/about

왜 이런 주제일까요?

ToC

NLP 그리고 TensorFlow Text, RaggedTensor

RaggedTensor
- RaggedTensor vs SparseTensor
- 자연어 처리와 RaggedTensor

유용한 tf.strings, tensorflow-text
- 한글과 Unicode
- 텍스트 처리에서의 tf.strings
- tensorflow-text의 tokenizer

NSMC로 TensorFlow Text와 RaggedTensor 맛보기

NLP 그리고, tf text, RaggedTensor

TensorFlow Text & RaggedTensor

RaggedTensor

RaggedTensor vs SparseTensor

https://www.tensorflow.org/guide/ragged_tensor

```
>>> tf.RaggedTensor.from_value_rowids(
... values=[3, 1, 4, 1, 5, 9, 2, 6],
... value_rowids=[0, 0, 0, 0, 2, 2, 2, 3])
<tf.RaggedTensor [[3, 1, 4, 1], [], [5, 9, 2], [6]]>
>>>
>>> tf.RaggedTensor.from_row_lengths(
... values=[3, 1, 4, 1, 5, 9, 2, 6],
... row_lengths=[4, 0, 3, 1])
<tf.RaggedTensor [[3, 1, 4, 1], [], [5, 9, 2], [6]]>
>>>
>>> tf.RaggedTensor.from_row_splits(
... values=[3, 1, 4, 1, 5, 9, 2, 6],
... row_splits=[0, 4, 4, 7, 8])
<tf.RaggedTensor [[3, 1, 4, 1], [], [5, 9, 2], [6]]>
```

자연어와 RaggedTensor

```
>>> model = tf.keras.Sequential([
... tf.keras.layers.Input(shape=[None], dtype=tf.int32, ragged=True),
... tf.keras.layers.Embedding(32, 16),
... tf.keras.layers.LSTM(16),
... tf.keras.layers.Dense(16, activation='relu'),
... tf.keras.layers.Dense(3, activation='softmax'),
... ])
>>> model(tf.ragged.constant([[1, 2, 3, 4], [1, 2, 3], [4, 5, 6, 7,
8, 9]]))
<tf.Tensor: shape=(3, 3), dtype=float32, numpy=
array([[0.33290705, 0.33258146, 0.3345115 ],
       [0.3322196 , 0.33285874, 0.3349216 ],
       [0.332826  , 0.33172128, 0.3354527 ]], dtype=float32)>
```

유용한 tf.strings, tensorflow-text

한글과 유니코드

"안녕하세요"

```
b'\xec\x95\x88\xeb\x85\x95\xed\x95\x98\xec\x84\xb8\xec\x9a\x94'
```

```
>>> string_tensor = tf.constant(["안녕하세요", "TensorFlow Everywhere Korea!",
"😊👋🤗🥕"])
>>>
>>> tf.strings.length(string_tensor)
<tf.Tensor: shape=(3,), dtype=int32, numpy=array([15, 28, 16],
dtype=int32)>
>>> tf.strings.length(string_tensor, unit='UTF8_CHAR')
<tf.Tensor: shape=(3,), dtype=int32, numpy=array([ 5, 28,  4],
dtype=int32)>
>>>
>>> tf.strings.substr(string_tensor, pos=0, len=1)
<tf.Tensor: shape=(3,), dtype=string, numpy=array([b'\xec', b'T', b'\xf0'],
dtype=object)>
>>> tf.strings.substr(string_tensor, pos=0, len=1, unit="UTF8_CHAR")
<tf.Tensor: shape=(3,), dtype=string, numpy=array([b'\xec\x95\x88', b'T',
b'\xf0\x9f\x98\x8a'], dtype=object)>
>>> [s.decode("UTF8") for s in tf.strings.substr(string_tensor, pos=0,
len=1, unit="UTF8_CHAR").numpy()]
['안', 'T', '😊']
```

```
>>> tf.strings.unicode_split(string_tensor, "UTF-8")
<tf.RaggedTensor [[b'\xec\x95\x88', b'\xeb\x85\x95', b'\xed\x95\x98',
b'\xec\x84\xb8', b'\xec\x9a\x94'], [b'T', b'e', b'n', b's', b'o', b'r',
b'F', b'l', b'o', b'w', b' ', b'E', b'v', b'e', b'r', b'y', b'w', b'h',
b'e', b'r', b'e', b' ', b'K', b'o', b'r', b'e', b'a', b'!'],
[b'\xf0\x9f\x98\x8a', b'\xf0\x9f\x91\x8b', b'\xf0\x9f\xa4\x97',
b'\xf0\x9f\xa5\x95']]>
>>> [[s.numpy().decode("UTF8") for s in v] for v in
tf.strings.unicode_split(string_tensor, "UTF-8")]
[['안', '녕', '하', '세', '요'], ['T', 'e', 'n', 's', 'o', 'r', 'F', 'l', 'o',
'w', ' ', 'E', 'v', 'e', 'r', 'y', 'w', 'h', 'e', 'r', 'e', ' ', 'K', 'o',
'r', 'e', 'a', '!'], ['😊', '👋', '🤗', '🥕']]
```

```
>>> text.normalize_utf8(['Äffin'])
<tf.Tensor: shape=(1,), dtype=string, numpy=array([b'\xc3\x84ffin'],
dtype=object)>
```

유용한 tf.strings, tensorflow-text

텍스트 처리에서의 tf.strings

- `tf.strings.split`
- `tf.strings.to_number`
- `tf.strings.strip`
- `tf.strings.regex_replace`

```
>>> tsv_rows = tf.constant([
... "6270596\t굳 ㅋ\t1",
... "9274899\tGDNTOPCLASSINTHECLUB\t0",
... "8544678\t뭐야 이 평점들은.... 나쁘진 않지만 10점 짜리는 더더욱 아니잖아\t0",
... ])
>>> splits = tf.strings.split(tsv_rows, sep='\t', maxsplit=2).to_tensor()
>>> string_inputs = tf.strings.strip(splits[:,1])
>>> string_inputs
<tf.Tensor: shape=(3,), dtype=string, numpy=
array([b'\xea\xb5\xb3 \xe3\x85\x8b', b'GDNTOPCLASSINTHECLUB',
       b'\xeb\xad\x90\xec\x95\xbc \xec\x9d\xb4
\xed\x8f\x89\xec\xa0\x90\xeb\x93\xa4\xec\x9d\x80....
\xeb\x82\x98\xec\x81\x98\xec\xa7\x84 \xec\x95\x8a\xec\xa7\x80\xeb\xa7\x8c
10\xec\xa0\x90 \xec\xa7\x9c\xeb\xa6\xac\xeb\x8a\x94
\xeb\x8d\x94\xeb\x8d\x94\xec\x9a\xb1
\xec\x95\x84\xeb\x8b\x88\xec\x9e\x96\xec\x95\x84'],
      dtype=object)>
>>> labels = tf.strings.to_number(splits[:,2], out_type=tf.int32)
>>> labels
<tf.Tensor: shape=(3,), dtype=int32, numpy=array([1, 0, 0], dtype=int32)>
```

```
>>> string_tensor = tf.constant(["안녕하세요 ㅎㅎㅎㅎㅎ", "안녕하세요!!!!!!"])
>>> string_tensor = tf.strings.regex_replace(string_tensor, "ㅎ{2,}", "ㅎㅎ")
>>> string_tensor = tf.strings.regex_replace(string_tensor, "!{2,}", "!!")
>>> [s.numpy().decode("UTF8") for s in string_tensor]
['안녕하세요 ㅎㅎ', '안녕하세요!!']
```

유용한 tf.strings, tensorflow-text

tensorflow-text의 Tokenizer

- `text.WhitespaceTokenizer`
- `text.UnicodeScriptTokenizer`
- `text.SentencepieceTokenizer`
- `text.BertTokenizer`
- …

유용한 tf.strings, tensorflow-text

tensorflow-text의 Tokenizer - WhitespaceTokenizer

```
>>> tokenizer = text.WhitespaceTokenizer()
>>> tokens = tokenizer.tokenize("안녕하세요! TensorFlow
Everywhere!")
>>> [s.decode("UTF8") for s in tokens.numpy()]
['안녕하세요!', 'TensorFlow', 'Everywhere!']
```

유용한 tf.strings, tensorflow-text

tensorflow-text의 Tokenizer - UnicodeScriptTokenizer

```
>>> tokenizer = text.UnicodeScriptTokenizer()
>>> tokens = tokenizer.tokenize("안녕하세요! TensorFlow
Everywhere!")
>>> [s.decode("UTF8") for s in tokens.numpy()]
['안녕하세요', '!', 'TensorFlow', 'Everywhere', '!']
```

https://unicode-org.github.io/icu-docs/

유용한 tf.strings, tensorflow-text

tensorflow-text의 Tokenizer - SentencepieceTokenizer

```
>>> tokenizer =
text.SentencepieceTokenizer(model=open('spm_model.model',
'rb').read())
>>> tokenizer.tokenize(['hello world']) # output type = tf.int32
...
>>> tokenizer =
text.SentencepieceTokenizer(model=open('spm_model.model',
'rb').read(), out_type=tf.string)
>>> tokenizer.tokenize(['hello world']) # output type =
tf.string
...
```

https://github.com/google/sentencepiece

유용한 tf.strings, tensorflow-text

tensorflow-text의 Tokenizer - BertTokenizer

"안녕하세요"

"안녕", "##하세요"

```
string_tensor_dataset = (
    tf.data ....
    .map( ...
) # string tensor만을 반환하는 dataset

from tensorflow_text.tools.wordpiece_vocab import bert_vocab_from_dataset
as bert_vocab

bert_vocab = bert_vocab.bert_vocab_from_dataset(
    string_tensor_dataset.batch(1000),
    vocab_size=8000,
    reserved_tokens=["<pad>", "<unk>", "<s>", "</s>"],
)

with tf.io.gfile.GFile("vocab.txt", "w") as out_file:
    for token in bert_vocab:
        print(token, file=out_file)
```

학습 준비

NSMC?

```
import io
import unicodedata

import sentencepiece as spm
import tensorflow as tf

def _get_nsmc_nfd():
    with open("nsmc/ratings.txt") as f:
        for line in f:
            yield unicodedata.normalize("NFD", line.split("\t")[1])

spm.SentencePieceTrainer.train(
    sentence_iterator=_get_nsmc_nfd(),
    model_prefix="spm",
    vocab_size=5000,
    normalization_rule_name="identity",
    pad_id=0,
    bos_id=1,
    eos_id=2,
    unk_id=3,
)
```

학습 준비

```
재밋는뎅 -> ['<s>', '_재미', 'ㅅ는데', 'ㅇ', '</s>']
애틋한 영화네요 -> ['<s>', '_', '애틋하', 'ㄴ', '_영화네요', '</s>']
```

NSMC로 TensorFlow Text와 RaggedTensor 맛보기

모델 학습

```
model = tf.keras.Sequential(
    [
        tf.keras.layers.Input(shape=[None], dtype=tf.int32, ragged=True),
        tf.keras.layers.Embedding(5000, 256),
        tf.keras.layers.LSTM(256),
        tf.keras.layers.Dense(256, activation="relu"),
        tf.keras.layers.Dense(2, activation="softmax"),
    ]
)
model.summary()
model.compile(optimizer="rmsprop",
loss=tf.keras.losses.SparseCategoricalCrossentropy(), metrics="acc")
```

```
with open("./spm.model", "rb") as spm_model:
    tokenizer = text.SentencepieceTokenizer(
        spm_model.read(),
        add_bos=True,
        add_eos=True)

def make_model_input(x: tf.Tensor) -> tf.Tensor:
    x = text.normalize_utf8(x, "NFD")
    return tokenizer.tokenize(x)
```

```
def parse_batch_tsv_rows(x: tf.Tensor) -> Tuple[tf.Tensor, tf.Tensor]:
    splits = tf.strings.split(x, sep="\t").to_tensor(shape=[tf.size(x), 3])
    model_inputs = make_model_input(splits[:, 1])
    labels = tf.strings.to_number(splits[:, 2])
    return model_inputs, labels

train_data = (
    tf.data.TextLineDataset("nsmc/ratings_train.txt")
    .skip(1)
    .shuffle(10000, reshuffle_each_iteration=True)
    .batch(64)
    .map(parse_batch_tsv_rows)
)
dev_data = train_data.take(100)
train_data = train_data.skip(100)

test_data = (
    tf.data.TextLineDataset(“nsmc/ratings_test.txt")
    .skip(1).batch(256).map(parse_batch_tsv_rows)
)
```

```
model.fit(train_data, validation_data=dev_data, epochs=3)
model.evaluate(test_data)
```

모델 학습 - 결과

```
2244/2244 [==============================] - 161s 71ms/step - loss: 0.4242 - acc: 0.8019 -
val_loss: 0.3179 - val_acc: 0.8612
Epoch 2/3
2244/2244 [==============================] - 160s 71ms/step - loss: 0.3174 - acc: 0.8646 -
val_loss: 0.3001 - val_acc: 0.8737
Epoch 3/3
2244/2244 [==============================] - 160s 71ms/step - loss: 0.2935 - acc: 0.8775 -
val_loss: 0.3034 - val_acc: 0.8759
196/196 [==============================] - 5s 24ms/step - loss: 0.3290 - acc: 0.8625
```

```
@tf.function(input_signature=tf.TensorSpec([None], dtype=tf.string))
def call(x: tf.Tensor) -> tf.Tensor:
    model_input = make_model_input(x)
    return model(model_input)

model.tokenizer = tokenizer
tf.saved_model.save(model, 'nsmc-model/0', call)
```

당근마켓

- 다양한 분야의 최신 기술을 적극적으로 도입하는 팀문화예요.
- 급격하게 올라가는 MAU와 많은 데이터, 글로벌화로 딥러닝을 효과적으로 적용할 재밌는 문제들이 많아요.

- 우리는
    - 현재는 GNN, Self-supervised 분야를 중요하게 생각하고 있어요.
    - 추천 시스템을 굉장히 중요하게 생각하며 적극적으로 개발 중이에요.
    - MLOps, 파이프라인, 인프라 작업도 많이 진행중이며, GCP를 적극적으로 활용해요

https://dngn.kr/join-us-dev

Q&A

감사합니다

https://github.com/jeongukjae/nsmc-tf-text
https://jeongukjae.github.io/posts/tensorflow-text-and-ragged-tensor/