TFRecords Dataset

Create a TFRecords dataset configured specifically for the project's dataset.

Chapter Goals:

- Create a TFRecords dataset for the input pipeline

A. Dataset from TFRecords file

After setting up the Example spec and feature parsing functions, we're finally ready to create TensorFlow datasets from the TFRecords files for both training and evaluation.

```
import tensorflow as tf

train_file = 'train.tfrecords'
eval_file = 'eval.tfrecords'
train_dataset = tf.data.TFRecordDataset(train_file)
eval_dataset = tf.data.TFRecordDataset(eval_file)
```

Parsing feature data from a serialized Example (ser_ex) using its corresponding Example spec (example_spec).

The TFRecords datasets contain serialized Example objects. Using the Example spec and feature parsing functions, we can convert each serialized Example to a tuple containing the Example's feature data and label.

```
import tensorflow as tf

example_spec = create_example_spec(True)
parse_fn = lambda ser_ex: parse_features(ser_ex, example_spec, True)
train_dataset = train_dataset.map(parse_fn)
eval_dataset = eval_dataset.map(parse_fn)
```

Using the functions from the previous two chapters to modify the TFRecords datasets

Using the functions from the previous two chapters to modify the TFRecords datasets.

The TFRecords dataset's `map` function allows us to apply the parsing function (`parse_fn`) to each serialized Example in the dataset. Since the `parse_features` function takes in two arguments, and `map` can only be used on functions with one argument, we use a single argument lambda function to wrap around `parse_features`.

B. Configuring the dataset

Now that we have the datasets for the input pipeline, we can decide how to configure them. Specifically, configuration refers to the shuffling, repetition, and batch size for the dataset.

For a refresher, recall the Configuration chapter from the previous section.

Shuffling datasets is always a good idea for training and evaluation, since it randomizes the order in which the data is passed into the machine learning model. The type of randomness depends on the buffer size for the shuffling.

We'll apply uniform random shuffling, which means that the buffer size needs to be at least the size of the dataset. In the **Preliminary Data Analysis** section, we calculated that the entire feature dataset contained 421570 rows. Therefore, we'll use a buffer size of 421570, which is guaranteed to be larger than the training and evaluation sets.

```
import tensorflow as tf

train_dataset = train_dataset.shuffle(421570)
eval_dataset = eval_dataset.shuffle(421570)
```

Uniformly randomly shuffling the datasets. Note that this is only applied during training/evaluation (i.e. when the dataset contains labels).

We also want to run training indefinitely, until we decide to kill the model running process manually (i.e. with `CTRL+C` or `CMD+C`). Evaluation is done with a single run-through of the dataset.

```
import tensorflow as tf
```

```
train_dataset = train_dataset.repeat()
```

Repeating the datasets indefinitely. The training will run until we manually kill the process.

Finally, we can set the dataset batch sizes, so that each training/evaluation step contains multiple data observations. Given the different sizes of the training and evaluation sets, we'll use batch sizes of 100 and 20, respectively.

```
import tensorflow as tf

train_dataset = train_dataset.batch(100)
eval_dataset = eval_dataset.batch(20)
```

Setting the batch size for the datasets.