Giriş

Önceki kısımda tokenizasyon işleminin tek bir cümle için ne kadar kolay olduğunu gördük. Ancak konu birden fazla cümleyle çalışmaya gelince bazı soru işaretleri ortaya çıkabilir.

- Multiple sequence veri ile nasıl başa çıkılabilir?
- Farklı uzunluktaki multiple sequencei le nasıl baş edilebilir?
- Vocab. indisleri, modelin iyi çalışmasındaki tek parametre midir?
- Çok uzun sequence'lar bir problem yaratır mı?

Girdi olarak batch kabul eden modeller

Önceki bölümlerde, bir ID listesini nasıl oluşturabileceğimizi gördük. Şimdi de bu listeleri tensor'e çevirerek modele vermeyi deneyebiliriz.

```
import torch
from transformers import AutoTokenizer, AutoModelForSequenceClassification

checkpoint = "distilbert-base-uncased-finetuned-sst-2-english"
tokenizer = AutoTokenizer.from_pretrained(checkpoint)
model = AutoModelForSequenceClassification.from_pretrained(checkpoint)

sequence = "I've been waiting for a HuggingFace course my whole life."

tokens = tokenizer.tokenize(sequence)
ids = tokenizer.convert_tokens_to_ids(tokens)
input_ids = torch.tensor(ids)
# This line will fail.
model(input_ids)
```

⚙️ Çıktı:

```
IndexError: Dimension out of range (expected to be in range of [-1, 0], but got
1)
```

Örnek kod çalıştırıldığında yukarıdaki gibi bir hata ile karşılaşılır. Bunun sebebi HuggingFace Transformer modelleri default olarak multiple sequence input kabul eder. Yukarıdaki işlemleri tokenizer'in her aşamasını manuel gerçekleştirerek ilerlettik. Normal durumda, `tokenizer()` direkt olarak çağrıldığında, aşağıdaki id'ler oluşturulur.

```
tokenized_inputs = tokenizer(sequence, return_tensors="pt")
tokenized_inputs["input_ids"]
```

⚙️ Çıktı:

```
tensor([[  101,  1045,  1005,  2310,  2042,  3403,  2005,  1037, 17662, 12172,
          2607,  2026,  2878,  2166,  1012,   102]])
```

Ancak modele verdiğimiz tensor aşağıdaki gibidir:

```
tensor([ 1045,  1005,  2310,  2042,  3403,  2005,  1037, 17662, 12172,  2607,
         2026,  2878,  2166,  1012])
```

Bu boyut farkından dolayı, kod hata vermiştir. Düzeltmek için aşağıdaki değişimi yapabiliriz:

```
#input_ids = torch.tensor(ids)
input_ids = torch.tensor([ids])
```

Batching, modele birden fazla cümle gönderme işlemidir ve bir adet cümle ile çalışmak kadar basittir. Fakat ikinci bir problem sözkonusudur. İki veya daha fazla cümleyi batch ettiğimizde bu cümlelerin uzunlukları birbirinden farklı olabilir. Buradaki problem tensor'lerin her durumda dikdörtgen olması gerekliliğidir. Bu yüzden farklı uzunluktaki (token uzunluğu) cümleleri direkt olarak tensor'lere çevirmek mümkün değildir. Buna çözüm olarak *padding* uygulanabilir.

Padding the inputs

Yukarıda bahsedilen nedenden dolayı aşağıdaki liste bir tensor'e çevrilemez:

```
batched_ids = [
    [200, 200, 200],
    [200, 200]
]
```

Padding uygulamak için bir `padding_token` tanımlanması gerekir. Cümle uzunluğu en uzun cümleye göre belirlendikten sonra daha kısa olan cümlelere padding uygulanabilir.

Tokenizer'a ait padding token id'ye ulaşabilmek için `tokenizer.pad_token_id` özelliğine bakılabilir. Aşağıda birbirinden farklı uzunlukta olan cümlelerin padding işlemi gerçekleştirilmiştir.

```
model = AutoModelForSequenceClassification.from_pretrained(checkpoint)

sequence1_ids = [[200, 200, 200]]
sequence2_ids = [[200, 200]]
batched_ids = [
    [200, 200, 200],
    [200, 200, tokenizer.pad_token_id],
]

print(model(torch.tensor(sequence1_ids)).logits)
print(model(torch.tensor(sequence2_ids)).logits)
print(model(torch.tensor(batched_ids)).logits)
```

⚙️ Çıktı:

```
tensor([[ 1.5694, -1.3895]], grad_fn=<AddmmBackward>)
tensor([[ 0.5803, -0.4125]], grad_fn=<AddmmBackward>)
tensor([[ 1.5694, -1.3895],
        [ 1.3373, -1.2163]], grad_fn=<AddmmBackward>)
```

Burada bir terslik vardır. Birinci ve ikinci cümlelerin logit'leri tekil ve batch olarak verildiğinde farklı logit değerleri üretilmiştir. Bunun sebebi, Transformer modellerinin verilen sequence'ı bir bağlamda incelemesidir. Eklediğimiz padding token'leri bu bağlamdan ayırmak için **attention_mask** tanımlanması gereklidir.

Attention masks

Attention mask'lar input id'lerden oluşan tensor ile aynı boyuttadır ve 0 veya 1 değerlerinden oluşur. Model tarafından görmezden gelinmesi istenilen token'lara karşılık gelen attention_mask tensor elementi, 0'a eşitlenir.

```
batched_ids = [
    [200, 200, 200],
    [200, 200, tokenizer.pad_token_id],
]

attention_mask = [
    [1, 1, 1],
    [1, 1, 0],
]

outputs = model(torch.tensor(batched_ids),
attention_mask=torch.tensor(attention_mask))
print(outputs.logits)
```

⚙️ Çıktı:

```
tensor([[ 1.5694, -1.3895],
        [ 0.5803, -0.4125]], grad_fn=<AddmmBackward>)
```

Bu sayede batch sequence çıktısı ile tekil sequence çıktısı aynı olur.

Longer sequences

Transformer modellerinde, input olarak verilen sequence için bir uzunluk kısıtı vardır. Çoğu model 512-1024 token uzunluğu ile başa çıkabilir, aksi takdirde de çalışmayı durdurur. İki çözümü olabilir:

1. Sequence'ları kısaltmak.
2. Daha uzun sequence'ları kabul edebilen modeller ile çalışmak.

Modellerin başarılı olabileceği sequence uzunluğu farklıdır ve bazı modeller uzun sequence'lar ile çalışmak için geliştirilmiştir. Örnek olarak **Longformer** veya **LED** verilebilir. Eğer iş daha uzun sequence'lar ile çalışmayı gerektiriyorsa bu iki model önerilebilir.

Aksi durumda, sequence'ları kırpmak bir seçenektir, aşağıdaki gibi gerçekleştirilebilir:

```
sequence = sequence[:max_sequence_length]
```