Uygulama 14b. Transformer Mimarisi İle NLP Uygulaması

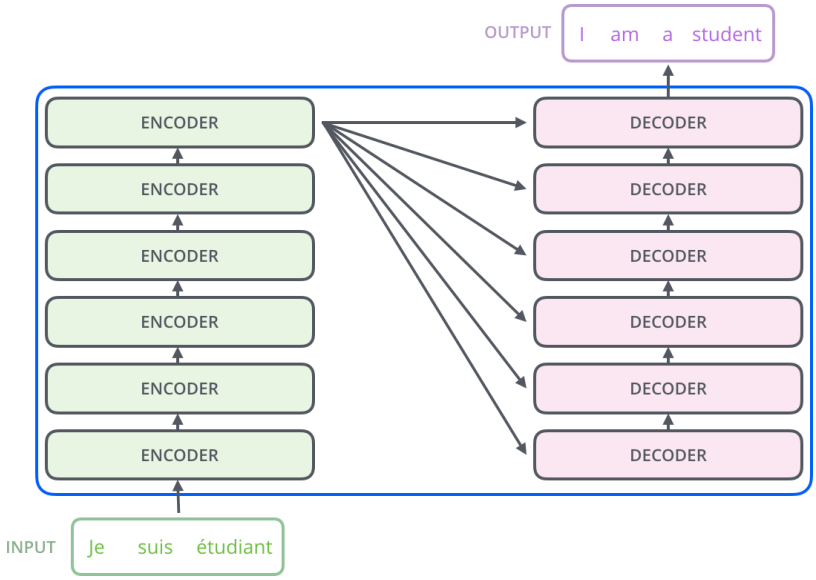
1. Amaç

Bu uygulamada Transformer BERT (Bidirectional Encoder Representations from Transformers) mimarisi ile yazılan bir cümleden duygu analizi işlemiyapılacaktır.

2. Teorik Bilgi

Doğal dil işleme (NLP), bilgisayarların insan dilini anlaması, yorumlaması ve üretmesiyle ilgili bir disiplindir. NLP, dilin karmaşıklıklarıyla başa çıkmak ve dildeki anlamı anlamak için bilgisayar bilimleri, yapay zeka ve dilbilim gibi alanlardan yararlanır. İşte NLP'nin temel konseptleri ve teorik bilgilerinden bazıları:

- **Tokenization (Belirteçleme):** Cümleleri, paragrafları veya metni daha küçük parçalara, yani "belirteçlere" bölmek anlamına gelir. Bu belirteçler genellikle kelime veya alt kelimeler olabilir.
- **Morfolojik Analiz:** Bir kelimenin yapısını inceleyen bu süreç, kelimenin kökünü ve eklerini belirlemeyi içerir. Bu, dilbilgisi kuralları ve dilin yapısal özelliklerini anlamak için önemlidir.
- **Sentiment Analysis (Duygu Analizi):** Metinlerin veya cümlelerin içerdikleri duygusal tonu belirlemek için kullanılan bir NLP uygulamasıdır. Genellikle pozitif, negatif veya nötr gibi kategorilere ayrılabilir.
- **Named Entity Recognition (NER):** Metindeki önemli bilgileri (örneğin, kişilerin adları, organizasyonların isimleri, tarihler, yerler) tanımlama sürecidir.
- **Part-of-Speech Tagging (POS):** Bir kelimenin dilbilgisel rolünü belirleme sürecidir. Örneğin, bir kelimenin bir isim, sıfat veya fiil olup olmadığını belirlemek.
- **Syntax (Sözdizimi):** Cümlelerin yapısal özelliklerini inceleyen bir alan. Bu, cümlelerdeki kelimelerin nasıl bir araya geldiğini ve bağlandığını anlamak için kullanılır.
- **Machine Translation (MT):** Bir dilde yazılmış metni başka bir dile çevirme sürecidir. Google Translate gibi çeviri hizmetleri bu tür teknikleri kullanır.
- **Word Embeddings:** Kelimeleri vektör uzayına gömmek, yani her kelimeyi bir sayısal vektörle temsil etmek. Bu, semantik benzerlikleri ölçmek ve kelime ilişkilerini anlamak için kullanılır.
- **Recurrent Neural Networks (RNN) ve Long Short-Term Memory (LSTM):** Dil modelleme ve metin oluşturma gibi NLP görevlerinde kullanılan derin öğrenme teknikleri. RNN ve LSTM gibi modeller, bir diziyi sıralı olarak işlemek ve zaman içindeki bağlantıları anlamak için tasarlanmıştır.
- **Transformer Architecture:** GPT (Generative Pre-trained Transformer) gibi modellerde kullanılan bir mimari türü. Büyük ölçekli dil modelleri oluşturmak için transfer öğrenme prensiplerini kullanır.



Resim 1. NLP Dil Çevirici Örneği

3. Uygulama

1. Model oluşturmak için gerekli kütüphanelerin ve veri setinin import edilmesi işlemi yapılmıştır.

```
#BERT (Bidirectional Encoder Representations from Transformers)

import torch
import pandas as pd
from tqdm.notebook import tqdm
```

```
[ ] df = pd.read_csv('/content/smile-annotations-final.csv',
                     names = ['id', 'text', 'category'])
    df.set_index('id', inplace=True)
```

```
[ ] df.text.iloc[100]
```

2. Verilerin sınıf dağılımlarını görmek için gerekli grafik kodlarının yazılması işlemi gerçekleştirilmiştir.

```
[ ] df.category.value_counts()


    # 'category'=['A','B','C','B','C']
    # df['category'].value_counts()
    # A:1 B:2 C:2
```

```
nocode               1572
happy                1137
not-relevant          214
angry                  57
surprise               35
sad                    32
happy|surprise         11
happy|sad               9
disgust|angry           7
disgust                 6
sad|disgust             2
sad|angry               2
sad|disgust|angry       1
Name: category, dtype: int64
```

```
[ ] df = df[~df.category.str.contains('\|')]
```

```
[ ] df = df[df.category != 'nocode']
```

3. Sınıfların her birine referans kod numaraları verilmesi işlemi gerçekleştirilmiştir.

```
[ ] possible_labels = df.category.unique()
```

```
[ ] label_dict = {}
    for index, possible_label in enumerate(possible_labels):
        label_dict[possible_label] = index
```

```
[ ] label_dict
```

```
{'happy': 0,
 'not-relevant': 1,
 'angry': 2,
 'disgust': 3,
 'sad': 4,
 'surprise': 5}
```

```
[ ] df['label'] = df.category.replace(label_dict)
```

```
[ ] df.head()
```

4. Veri setinin %20'si test olacak şekilde parçalanma işlemi gerçekleştirilmiştir.

```
[ ] X_train, X_val, y_train, y_val = train_test_split(df.index.values, df.label.values, test_size = 0.20, random_state=17, stratify = df.label.values)
```

```
[ ] df['data_type'] = ['not_set']*df.shape[0]
```

```
[ ] df.head()
```

id	text	category	label	data_type
614484565059596288	Dorian Gray with Rainbow Scarf #LoveWins (from...	happy	0	not_set
614746522043973632	@SelectShowcase @Tate_StIves ... Replace with ...	happy	0	not_set
614877582664835073	@Sofabsports thank you for following me back. ...	happy	0	not_set
611932373039644672	@britishmuseum @TudorHistory What a beautiful ...	happy	0	not_set
611570404268883969	@NationalGallery @ThePoldarkian I have always ...	happy	0	not_set

```
[ ] df.loc[X_train, 'data_type'] = 'train'
    df.loc[X_val, 'data_type'] = 'val'
```

5. Transformer kütüphanesi import edilip, Bert ön eğitimli model eklenmiştir.

```
[ ] from transformers import BertTokenizer
    from torch.utils.data import TensorDataset
```

```
[ ] tokenizer = BertTokenizer.from_pretrained('bert-base-uncased', do_lower_case=True)
```

```
Downloading (…)okenizer_config.json: 100% 28.0/28.0 [00:00<00:00, 1.57kB/s]
Downloading (…)solve/main/vocab.txt: 100% 232k/232k [00:00<00:00, 10.0MB/s]
Downloading (…)/main/tokenizer.json: 100% 466k/466k [00:00<00:00, 15.6MB/s]
Downloading (…)lve/main/config.json: 100% 570/570 [00:00<00:00, 10.1kB/s]
```

```
[ ] encoded_data_train = tokenizer.batch_encode_plus(df[df.data_type=='train'].text.values,
                                                     add_special_tokens=True,return_attention_mask=True,
                                                     pad_to_max_length=True,max_length=256,return_tensors='pt')
```

```
Truncation was not explicitly activated but `max_length` is provided a specific value, please use `truncation=True` to explicitly truncate examples to max
/usr/local/lib/python3.10/dist-packages/transformers/tokenization_utils_base.py:2614: FutureWarning: The `pad_to_max_length` argument is deprecated and wil
  warnings.warn(
```

```
[ ] df.head()
```

6. Eğitimi başlatan kod eklenmiş ve eğitim başlatılmıştır.

```
[ ]     for batch in progress_bar:

            model.zero_grad()

            batch = tuple(b.to(device) for b in batch)

            inputs = {
                'input_ids' : batch[0],
                'attention_mask' : batch[1],
                'labels' : batch[2]

            }

            outputs = model(**inputs)

            loss = outputs[0]
            loss_train_total += loss.item()
            loss.backward()

            torch.nn.utils.clip_grad_norm_(model.parameters(), 1.0)

            optimizer.step()
            scheduler.step()

            progress_bar.set_postfix({'training_loss' : '{:.3f}'.format(loss.item()/len(batch))})

        #torch.save(model.state_dict(), f'Models/BERT_ft_epoch{epoch}.model')

        tqdm.write('\nEpoch {epoch}')

        loss_train_avg = loss_train_total/len(dataloader_train)
        tqdm.write('Training loss: {loss_train_avg}')
```

4. Uygulama Soruları:

1) Verilen kodu inceleyerek farklı bir ön eğitimli model kullanarak kendi eğitimizini gerçekleştiriniz.
2) Verilen kodlarda eğitilen modelin karmaşıklık matrisi ve ROC eğrisini çizecek kodu yazınız ve bu görselleri yorumlayınız.
3) Model ile accuracy, F1-Score ve loss değerlerini kontrol ediniz. Modelin kayıp ve accuracy grafiklerini çizdiriniz.
4) Aynı modeli farklı bir veri seti ile eğitip sonuçlarınızı tartışınız.

Veri Seti Linki: https://www.kaggle.com/datasets/ashkhagan/smile-twitter-emotion-dataset