Image Captioning

Nihal Demir

19010011016

Dikkat Mekanizması Nedir?

Attention

Attention (dikkat) kelime anlamı olarak, bir veya bir kaç şeye odaklanırken geri kalan şeylerin ihmal edilmesi anlamına gelir. Bir sinir ağı, insan beyni hareketlerinin sadeleştirilmiş şekilde çalışması üzerinde temellendirilmiştir. Attention mekanizması da sinir ağına benzer şekilde, benzer olan şeylere odaklanarak, geri kalan şeylerin filtrelenmesi, yok sayılması temeline dayandırılmaktadır.

Dikkat mekanizması, NLP'de encoder-decoder temelli sinirsel makine çeviri sisteminde bir gelişme olarak ortaya çıkmıştır. Daha sonra bu mekanizma ve benzerleri computer vision veya speech processing gibi farklı uygulamalarda da kullanılmıştır.

Dikkat Mekanizması

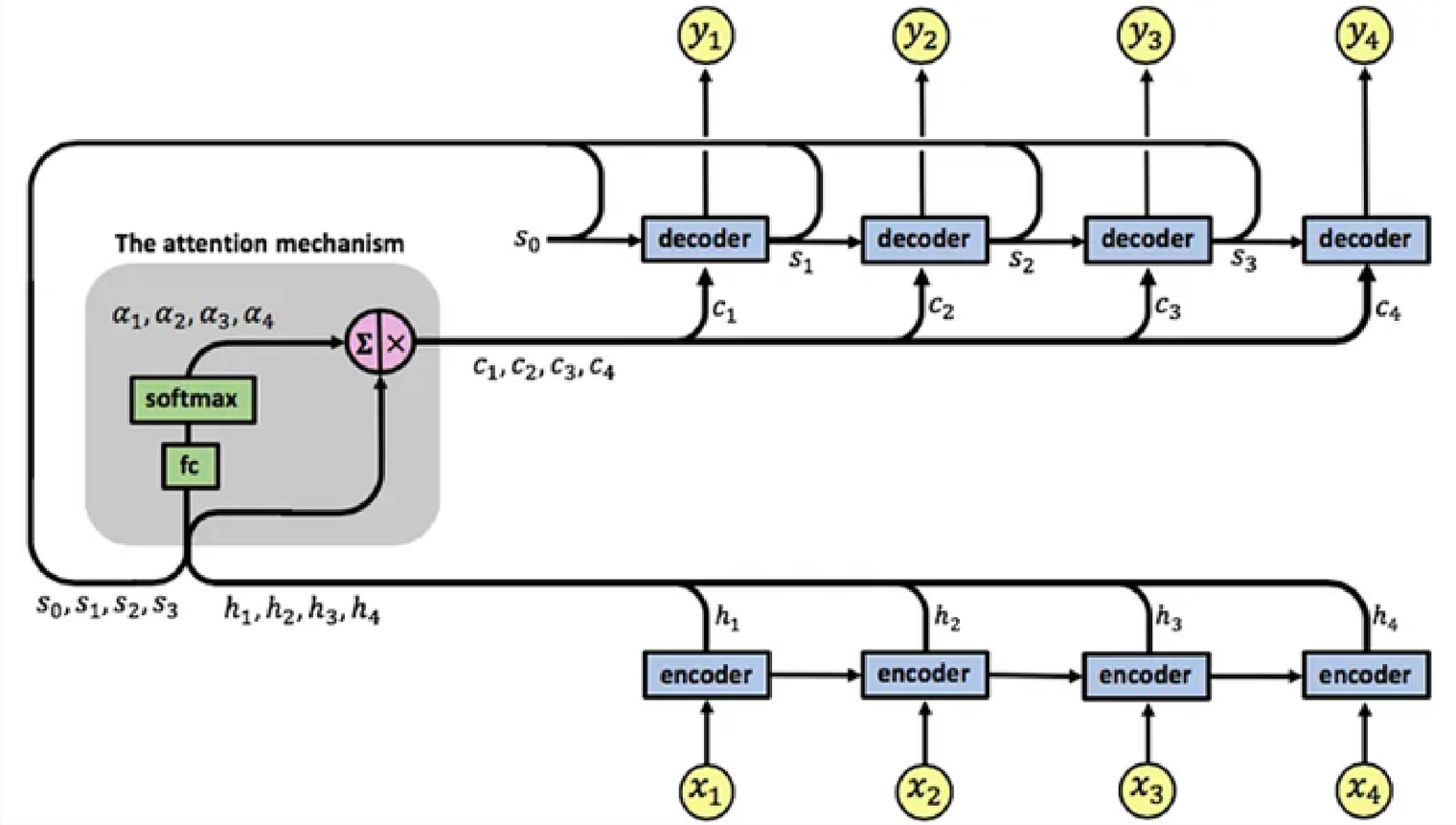
Nasıl Çalışır?

Dikkat mekanizması RNN'de birleştirilen, çıktı dizisinin belirli bir bölümünü tahmin ederken girdi dizisinin belirli bölümlerine odaklanılmasını sağlayan, daha kolay öğrenmeyi ve daha yüksek kaliteli çıktılar sağlayan bir mekanizmadır.

Dikkat mekanizması kodlayıcı ve kod çözücü arasında yer almaktadır. Girdi olarak kodlayıcının çıktı vektörleri alınırken çıktı olarak bağlam (context) vektörü olarak adlandırılan bir dizi vektör verir. Bağlam vektörleri, kod çözücünün çıktısını tahmin ederken girdinin belirli bölümlerine odaklanmasını sağlayan vektörlerdir. Her bağlam vektörü, kodlayıcının çıkış vektörlerinin ağırlıklı bir toplamıdır.

Genellikle bağlam vektörleri ile hesaplanan dikkat ağırlıkları, bir dikkat mekanizması kullanılarak elde edilen ve bir dizi girdiye ne kadar önem verildiğini ifade eden bir değerler vektörüdür.Dikkat ağırlıkları, genellikle softmax fonksiyonuyla normalleştirilir, böylece toplam ağırlıklar 1'e eşit olur. Bu ağırlıklar, dikkatli olunması gereken giriş vektörlerinin göreceli önemini temsil eder.

Dikkat Mekanizması



Modelin Oluşturulması

Model

4 mantıksal bileşenden oluşmaktadır:

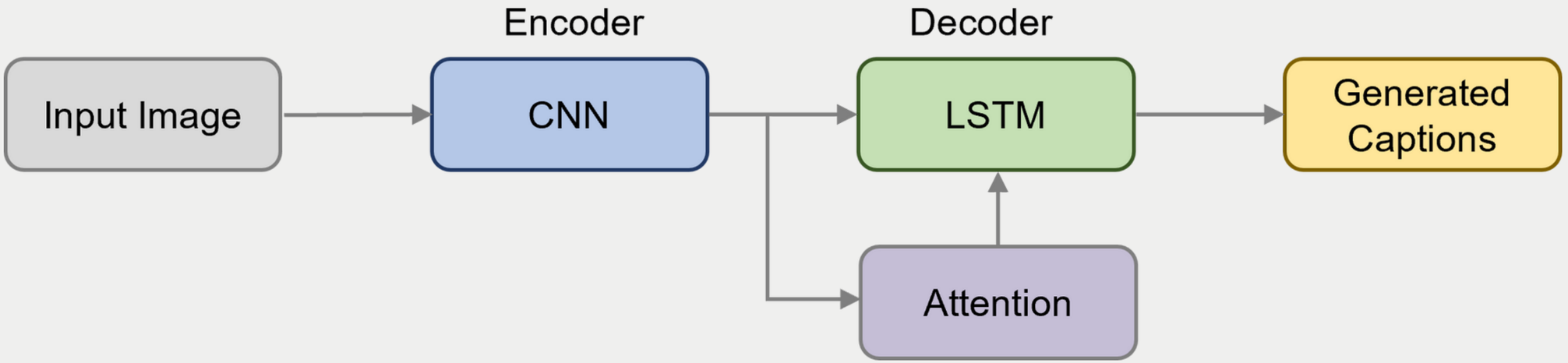
1. **Encoder:** Özellik çıkarım işlemi önceden eğitilmiş VGG16 modeli tarafından yapılmıştır. Bu bileşende kodlanmış (encoded) görüntü özelliklerini alan ve bunları kod çözücüye (decoder) ileten bir doğrusal (linear) katman bulunur.

2. **Sequence Decoder:** GRU ile oluşturulmuş tekrarlayan bir ağdır. Cümleler, gömme (Embedding) katmanından geçtikten sonra girdi olarak iletilir.

3. **Attention:** Decoder, çıktı dizisinin her bir kelimesini üretirken, Dikkat modülü, söz konusu kelimeyi oluşturmak için görüntünün en ilgili kısmına odaklanmasına yardımcı olur.

4. **Cümle Oluşturucu:** Bu modül bir çift doğrusal katmandan oluşur. Kod çözücüden çıktı alır ve tahmin edilen dizideki her konum için sözlükteki her kelime için bir olasılık üretir.

Model


Encoder
Decoder
Input Image
CNN
LSTM
Generated Captions
Attention

Kodlama Aşaması

Parametreler

Buffer size, embedding, units gibi parametreler bir önceki model ile karşılaştırabilmek adına aynı değerlerde ayarlandı.

attention_features_shape, dikkat mekanizmasının kullanıldığı katmanlarda kullanılan girdi özelliklerinin şeklini temsil eder. VGG-16'dan elde edilen özelliklerin boyutu genellikle 7x7 olduğundan, attention_features_shape değeri 49 olarak ayarlanmıştır.

Parametreler

```
BATCH_SIZE = 16
BUFFER_SIZE = 1000
embedding_dim = 256
units = 512
vocab_size = len(tokenizer.word_index) + 1
num_steps = len(img_name_train) // BATCH_SIZE
features_shape = 512
attention_features_shape = 49
```

Encoder

VGG16 Encoder sınıfı, özellik çıktılarını Tam Bağlantılı (Fully Connected) katmandan geçirir. Özellik çıktıları belirlenen boyuta indirgenerek embedding işlemi gerçekleştirilir. Embedding, bir dilin veya verilen verideki kelimelerin tek tek, daha az boyutlu bir uzayda gerçek değerli vektörler olarak ifade edilmesidir.

Embedding katmanı, girdi bilgisini yüksek boyutlu uzaydan düşük boyutlu uzaya eşler ve ağın girdiler arasındaki ilişki hakkında daha fazla şey öğrenmesine ve verileri daha verimli bir şekilde işlemesine olanak tanır.
call fonksiyonu, sınıfın çağrılabilir bir nesne olmasını sağlar. Bu yöntemde, x parametresiyle gelen girdi özelliklerinin üzerinden geçilir. ReLU aktivasyon fonksiyonu kullanılarak aktivasyon uygulanır ve son çıktı döndürülür.

VGG16 Encoder

```python
class VGG16_Encoder(tf.keras.Model):
    def __init__(self, embedding_dim):
        super(VGG16_Encoder, self).__init__()
        self.fc = tf.keras.layers.Dense(embedding_dim)
        self.dropout = tf.keras.layers.Dropout(0.5, noise_shape=None, seed=None)

    def call(self, x):
        #x= self.dropout(x)
        x = self.fc(x)
        x = tf.nn.relu(x)
        return x
```

Decoder

RNN Decoder

```
class Rnn_Local_Decoder(tf.keras.Model):
 def __init__(self, embedding_dim, units, vocab_size):
   super(Rnn_Local_Decoder, self).__init__()
   self.units = units
   self.embedding = tf.keras.layers.Embedding(vocab_size, embedding_dim)
   self.gru = tf.keras.layers.GRU(self.units,
                                  return_sequences=True,
                                  return_state=True,
                                  recurrent_initializer='glorot_uniform')

   self.fc1 = tf.keras.layers.Dense(self.units)

   self.dropout = tf.keras.layers.Dropout(0.5, noise_shape=None, seed=None)
   self.batchnormalization = tf.keras.layers.BatchNormalization(axis=-1, momentum=0.99, epsilon=0.001, center=True, scale=True, beta_initializer='zeros', gamma_initializer='ones',
                                                                 moving_mean_initializer='zeros', moving_variance_initializer='ones', beta_regularizer=None, gamma_regularizer=None, beta_constraint=None, gamma_constraint=None)

   self.fc2 = tf.keras.layers.Dense(vocab_size)
```

GRU (Gated Recurrent Unit) hücreleri kullanılarak RNN Decoder modeli tanımlandı.
__init__ fonksiyonu içerisinde embedding, GRU, dense katmanı gibi gerekli katmanlar oluşturuldu ve modelin başlatılması sağlandı.

Decoder

- self.Uattn ve self.Wattn: Attention mekanizmasında kullanılan Uattn ve Wattn dense katmanları. Bu katmanlar, gizli durumu (hidden state) özellik uzayına dönüştürmek için kullanılır.

- self.Vattn: Bu katman, atama skorlarını hesaplamak için kullanılır. Skorlar, tanh aktivasyon fonksiyonu ile gizli durum ve özelliklerin birleşiminden geçirildikten sonra bu katmandan geçirilir.

Dense, katmanlar arasında nöron ya da düğümlerin geçişlerini sağlar. Bir başka deyişle, bir katmandan aldığı nöronları bir sonraki katmana girdi olarak bağlanmasını sağlar.

Bu üç katman attention mekanizmasını oluştmaktadır. Attention mekanizması, gizli durumun belirli bir ağırlıkla özelliklerin farklı kısımlarına odaklanmasını sağlar. Bu, çıktıyı oluştururken önemli olan özelliklere daha fazla dikkat verilmesini sağlar.

Attention

```
# Implementing Attention
self.Uattn = tf.keras.layers.Dense(units)
self.Wattn = tf.keras.layers.Dense(units)
self.Vattn = tf.keras.layers.Dense(1)
```

U, V ve W genellikle attention mekanizmasında kullanılan dense katmanlara atanırken tercih edilen genel bir adlandırma şeklidir. "U" genellikle gizli durumu temsil ederken, "W" özellikleri temsil eder ve "V" atama skorlarını temsil eder. Bu tür adlandırmalar, attention mekanizmasının nasıl çalıştığına ilişkin bir geleneksel adlandırma uygulamasıdır.

Decoder

call fonksiyonu ile RNN decoderın dikkat mekanizmasını, gömülü girdiyi ve gizli durumu kullanarak bir çıktı ve güncellenmiş gizli durumu hesaplaması sağlandı.

Verilen girdi (x), dikkate alınan özellikler (features) ve gizli durum (hidden) üzerinde çalışır. Bağlam vektörünü hesaplamak için dikkat mekanizması kullanılır. Bağlam vektörünü hesaplama aşamasında tanh ve ağırlıklandırma işlemleri gerçekleştirilir. Ağırlıklar softmax fonksiyonu kullanılarak normalleştirilir. Girdi dizisi embedding işlemine tabi tutulur, gömülür. Bağlam vektörü ile gömülü girdi birleştirilir.

GRU katmanı kullanılarak çıktı ve güncellenmiş gizli durum elde edilir. Çıktı tensörü, tam bağlantı katmanına verilerek gizli temsil elde edilir. Tensörün boyutu yeniden şekillendirilir. Dropout ve batch normalization katmanları uygulanır. Son olarak, tam bağlantı katmanı kullanılarak kelime dağılımı elde edilir.

```
def call(self, x, features, hidden):

  hidden_with_time_axis = tf.expand_dims(hidden, 1)
  # score shape == (64, 49, 1)
  # Attention
  '''e(ij) = f(s(t-1),h(j))'''
  ''' e(ij) = Vattn(T)*tanh(Uattn * h(j) + Wattn * s(t))'''

  score = self.Vattn(tf.nn.tanh(self.Uattn(features) + self.Wattn(hidden_with_time_axis)))
  attention_weights = tf.nn.softmax(score, axis=1)

  context_vector = attention_weights * features
  context_vector = tf.reduce_sum(context_vector, axis=1)

  x = self.embedding(x)
  x = tf.concat([tf.expand_dims(context_vector, 1), x], axis=-1)

  output, state = self.gru(x)
```

```
  x = self.fc1(output)
  x = tf.reshape(x, (-1, x.shape[2]))

  x= self.dropout(x)
  x= self.batchnormalization(x)

  x = self.fc2(x)

  return x, state, attention_weights

def reset_state(self, batch_size):
  return tf.zeros((batch_size, self.units))


encoder = VGG16_Encoder(embedding_dim)
decoder = Rnn_Local_Decoder(embedding_dim, units, vocab_size)
```

Evaluate

Evaluate fonksiyonu decoder'i kullanarak görüntüye dayalı metin açıklamasını oluşturur. Her adımda decoder girdi kelimesini tahmin eder ve dikkat mekanizması ile decoderin hangi özelliklere odaklandığı belirlenir.

Yerel çözücünün gizli durumu sıfırlandı. Verilen görüntüden özellik çıkarımı yapıldı ve bu özellikler reshape fonksiyonu ile uygun şekle dönüştürüldü. Özellikler, encoder modeline verilerek gizli durum ve dikkate alınacak özellikler elde edildi.

Decoder'e mevcut girdi, dikkate alınan özellikler ve gizli durum verilerek tahminler, güncellenmiş gizli durum ve dikkat ağırlıkları elde edildi. Dikkat ağırlıkları, uygun şekle dönüştürülerek başlangıçta np.zeros ile oluşturulan attention plot matrisine eklendi. Fonksiyondan result ve attention plot değerleri döndürüldü.

```python
def evaluate(image):
    attention_plot = np.zeros((max_length, attention_features_shape))

    hidden = decoder.reset_state(batch_size=1)
    temp_input = tf.expand_dims(load_image(image)[0], 0)
    img_tensor_val = image_features_extract_model(temp_input)
    img_tensor_val = tf.reshape(img_tensor_val, (img_tensor_val.shape[0], -1, img_tensor_val.shape[3]))

    features = encoder(img_tensor_val)
    dec_input = tf.expand_dims([tokenizer.word_index['']], 0)
    result = []

    for i in range(max_length):
        predictions, hidden, attention_weights = decoder(dec_input, features, hidden)
        attention_plot[i] = tf.reshape(attention_weights, (-1, )).numpy()
        predicted_id = tf.argmax(predictions[0]).numpy()
        result.append(tokenizer.index_word[predicted_id])

        if tokenizer.index_word[predicted_id] == '':
            return result, attention_plot

        dec_input = tf.expand_dims([predicted_id], 0)
    attention_plot = attention_plot[:len(result), :]

    return result, attention_plot
```

Sonuçlar

1

Real Caption:

- Black dog and spotted dog are fighting.
- Black dog and tri-colored dog playing with each other on the road.
- Black dog and white dog with brown spots are staring at each other in the street.
- Two dogs of different breeds looking at each other on the road.
- Two dogs on pavement moving toward each other.

Prediction:

- Dog is playing with each other.
- BLEU score: 0.076

2

Real Caption:

- Little girl covered in paint sits in front of painted rainbow with her hands in bowl.
- Little girl is sitting in front of large painted rainbow.
- Small girl in the grass plays with fingerpaints in front of white canvas with rainbow on it.
- There is girl with pigtails sitting in front of rainbow painting.
- Young girl with pigtails painting outside in the grass.

Prediction:

- Young girl in red dress is sitting in front of rainbow painting.
- BLEU score: 0.016

3

Real Caption:

- Man in hat is diplaying pictures next to skier in blue hat.
- Man skis past another man displaying paintings in the snow.
- Person wearing skis looking at framed pictures set up in the snow.
- Skier looks at framed pictures in the snow next to trees.
- Man on skis looking at artwork for sale in the snow.

Prediction:

- Person skis in the skier in black hat and hat and woman.
- BLEU score: 0.039

★ Teşekkürler ★