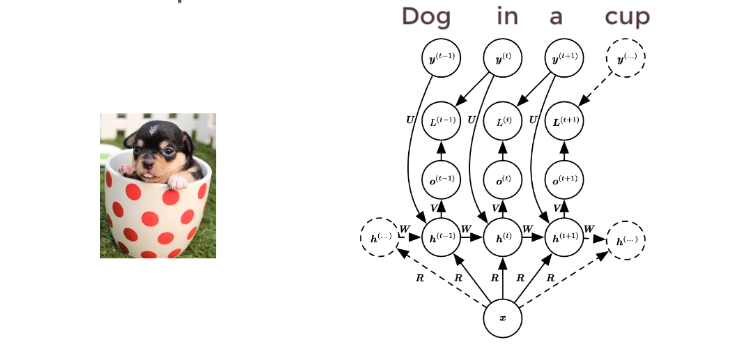
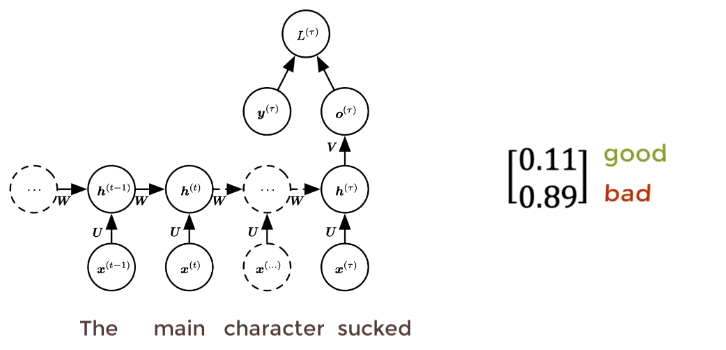
# Recurrent Neural Networks (Sıralı Veri)

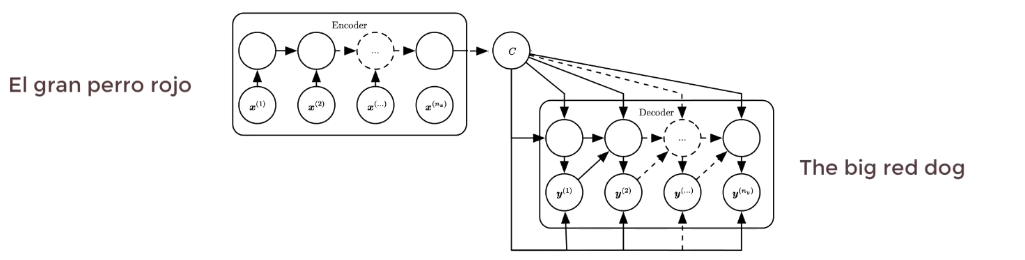
## Vector-Sequence Modeller

Sabit boyutlu bir vektörü girdi olarak alır ve herhangi bir uzunlukta sıralı çıktı sunar. Bu duruma örnek olarak verilen bir resmin çıktısının, resmi anlatan kelimeler olduğu düşünülebilir. Koşan bir köpek fotoğrafı verilen model, çıktı olarak “Köpek kırlarda koşuyor.” cümlesini verebilir.

## Sequence-Vector Modeller

Sıralı bir girdi alır ve çıktı olarak istenen uzunlukta bir vektör sunar. Duygusallık analizi gibi işlemlerde kullanılabilir. Örnek olarak, bir film hakkında yazılmış “Ana karakter çok kötüydü.” yorumunu girdi olarak alan bir Sequence-Vector model, (isteğe bağlı olarak) çıktısında yorumu yapan kişinin film hakkında düşüncesini [İyi, Kötü] verebilir.

## Sequence-Sequence Modeller

En popüler olan modellerdir. Sıralı bir girdi alır ve sıralı bir çıktı sunar. Genel anlamda en fazla, dil çevirilerinde kullanılır (Google Translate). İspanyolca verilen bir cümleyi İngilizce’ye dönüştürebilir.

## RNN’lerin Dezavantajları

Geleceği tahmin etme noktasında sıkça kullanılsalar da bazı noktalarda RNN’lerin geliştirilmeye ihtiyacı var. RNN’lerin dezavantajlarından bahsetmek gerekirse;

* Veri eğitme noktasında yavaş. Bu yavaşlık o kadar fazla ki kullanılırken geri yayılım kesilmiş şekilde kullanılıyor. Buna rağmen tamamen donanıma bağlı bir hız durumu söz konusu.
* Uzun sıralı girdilerde iyi çalışmıyor. Eğer ağlar uzunsa vanishing (yok olma) ya da gradyan patlama (exploding gradients) durumları oluşuyor.

Bu durumları engellemek ve değiştirmek içinse RNN’ler üzerinde farklı NN’ler geliştirildi.

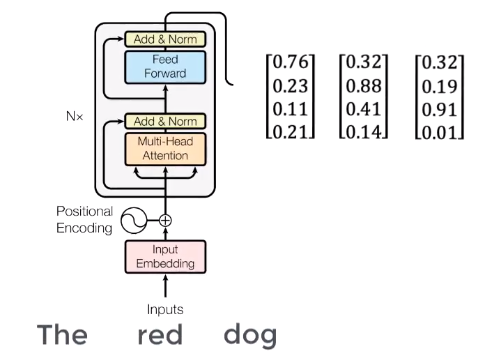
# LSTM Networks

saat, çizim içeren bir resim

Açıklama otomatik olarak oluşturulduRNN’i geliştirmek ve uzun için **LSTM** (Long short term memory) ortaya çıktı. LSTM sayesinde bilgilerin uzun sıralı girdilerde dahi tutulabilmesi sağlandı. Vanishing ve exploding gradient problemlerinin önüne geçildi. Bunu da modele eklenen yeni bir branch/dal ile sağladı. Bu branch sayesinde gerekli durumlarda işlemlerin tekrarlanmayıp, yapılmadan sonraki hücreye (cell) geçilebilmesi gerçekleştirildi.

Fakat böyle bir branch ve farklı daha birçok şeyin eklenmesi, LSTM’leri daha kompleks bir hale getirdiği için **RNN’lerden dahi daha yavaş** olmasına neden oldu. Bunun yanında sıralı ve birbirine bağlı olarak girdiyi aldığı için LSTM’i default olarak paralel şekilde çalıştırmak mümkün değil. Bunu yapabilmek için yakın zamanda ortaya çıkan bir kavrama (**Transformer**) ve getirdiklerine bakmamız gerekiyor.

# Transformer

Transformer neural network mimarisi, 2017 yılında ortaya çıktı. Encoder ve Decoder’lara bakıldığında daha çok Recurrent Neural Network’lere benzediği söylenebilir. Farkı ise sıralı olan girdi NN içerisine paralel olarak aktarılabilir.

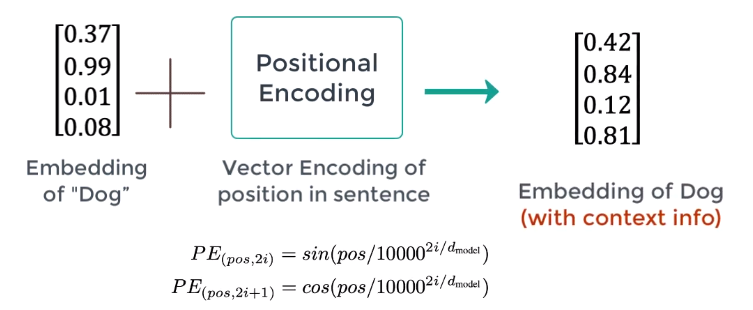
Örnek olarak 3 kelimeden oluşan bir cümleyi Türkçe’den İngilizce’ye çevirmek istersek RNN bunu gerçekleştirmek için önce ilk kelimeyi alıp karşılığını, ardından ikinci kelimeyi alıp karşılığını ve son olarak da üçüncü kelimeyi alıp karşılığını bulacak ve sonucu bize döndürecektir. Sıralı yaptığı için de bunu gerçekleştirmesi epey bir zaman alacaktır. Fakat zamana bağlı bir durum söz konusu olmadığı için, bunu **Transformer** ile yapmak istersek, üç kelime de aynı anda modele alınacak ve paralel olarak olasılıkları hesaplanıp sonuçları döndürülecektir. RNN’den neredeyse 3 kat daha hızlı olacağı düşünülebilir.

## Transformer Bileşenleri

### Input Embedding

Makineler kelimeleri anlayamadığı için sayı, vektör ya da matrise çevrilmesi gerekiyor. Bu map oluşturma olayı, benzer, zıt ya da aynı anlama gelen kelimelerin aynı kümeye konulması ile başlıyor. Kelimeler arasındaki uzaklık (cost) **Embedding Space** olarak isimlendiriliyor. Bununla alakalı hali hazırda eğitilmiş modelleri bulabilmek mümkün. Embedding space’e verilen bir kelime, vektör olarak çıkıyor. Fakat bu kelimelerin aynı cümlede farklı yerlerde bulunması farklı anlamlar ortaya çıkaracağı için burada karşımıza **Positional Encoder** kavramı çıkıyor.

### Positional Encoder

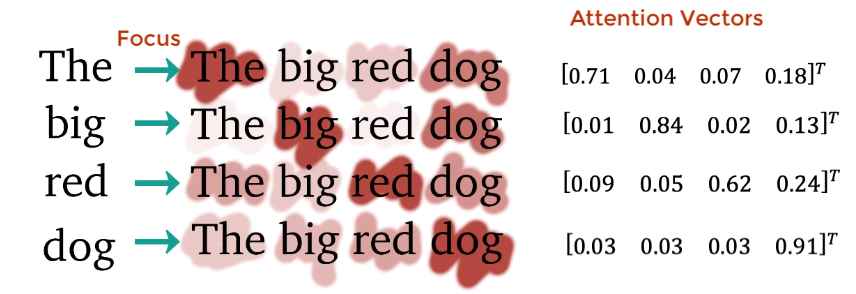
Positional Encoder, kelimenin cümledeki yeri ile alakalı oluşan bir vektördür. Input embedding ile birleştirildiğinde karşımıza gerçek vektör çıktısı gelecektir.

Input embedding ve positional encoder bileşenlerinden sonra ise karşımıza çıkan bileşen biraz daha karmaşık ve Transformer’in kendisini oluşturan, Encoder Block kısmı.

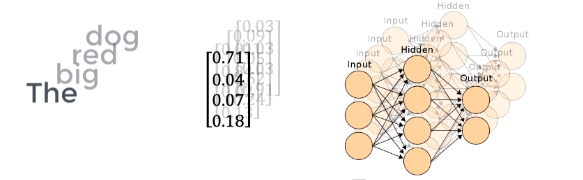
### Encoder Block: Multi-Head Attention

metin, harita içeren bir resim

Açıklama otomatik olarak oluşturulduBurada karşımıza çıkan ve karmaşaya sebep olabilecek kelime “**attention**”. Attention NN içerisinde, girdide hangi kısma odaklanmalıyım sorunun cevabıdır. **Soft** ve **Hard** olarak ayrılır. Eğer birden fazla noktaya odaklanılacaksa Soft, tek bir noktaya odaklanılacaksa Hard Attention kullanılır.

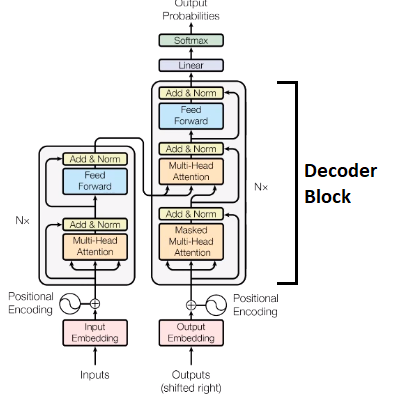
İngilizceden Türkçe’ye çeviri yapmak istediğimizde kelimeleri paralel olarak işlemenin yanında, kelimelerin birbirileri ile olan ilişkileri ve neye ne kadar önem verileceği de önemlidir. Attention ile cümle içerisinde odaklanılacak noktanın iyi seçilmesi modelin doğru bir çeviri yapmasını da sağlayacaktır. Verilen cümlede 4 kelime olduğunu varsayarsak ve 4 kelime için de farklı Attention vektörlerin oluşturulduğunu düşünürsek her biri için kelimeler arasındaki ilişkiler farklı olacaktır.

### Encoder Block: Feed Forward

Basit RNN’dir. Her bir Attention vector için çalışır. Attention vector’lerin sonraki encoder block veya decoder block için anlaşılabilir hale (digestible) gelmesini sağlar.

### Decoder Block: Attention

Encoder block’a benzer bileşenleri bulunur fakat amacı girdiyi makinenin işleyebileceği hale getirmek değil, makinenin işlediği bilgiyi bizim anlayabileceğimiz hale getirmektir.

Decoder block, Positional Encoding ile Linear bileşenleri arasındaki kısımdır. Encoder’dan farklı olarak bulunan Masked Multi-Head Attention bileşeni çıktı olarak oluşacak vektör için, “Self Attention” ile cümlelerin Attention vector’lerini oluşturur. Ardından Encoder ve Decoder Attention vector’leri “Encoder-Decoder Attention” diyebileceğimiz bir blokta işlenir. Bu blokta her bir kelime vektörünün birbirleriyle ne kadar ilişki olduğu belirlenecek, girdi ve çıktıdaki her bir eleman için Attention vector’leri çıktı olarak oluşturacaktır. Her ikisinde de elemanların, diğer elemanlarla olan ilişkileri temsil edilecektir.

### Decoder Block: Feed Forward

Çıkış vektörünü bir sonraki decoder bloğu ya da Linear Layer tarafından daha anlaşılabilir hale getirir.

### Linear Layer

Linear Layer, garip gelebilir fakat bir Feed Forward Connected Layer’dir. Çeviri örneği için her bir nöron çıktı olarak kullanılacak dildeki kelimelerdir.

### Softmax

Softmax Layer, model çıktısını insan tarafından yorumlanabilen Probability Distribution çıktısına çevirir. Son olarak da en yüksek olasılığa sahip olan eleman çıktı olarak oluşur.

Bu işleyiş tek bir eleman için yapıldığından, sonraki elemanlar için de paralel olarak yapılır.