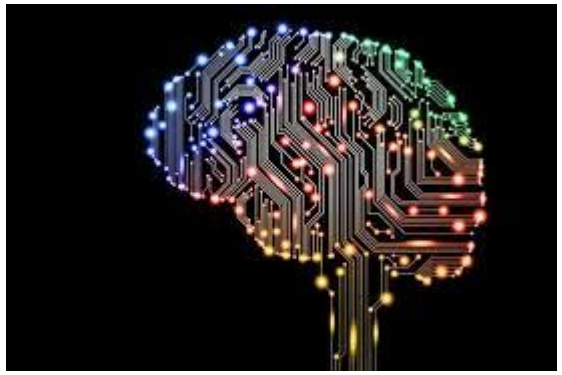
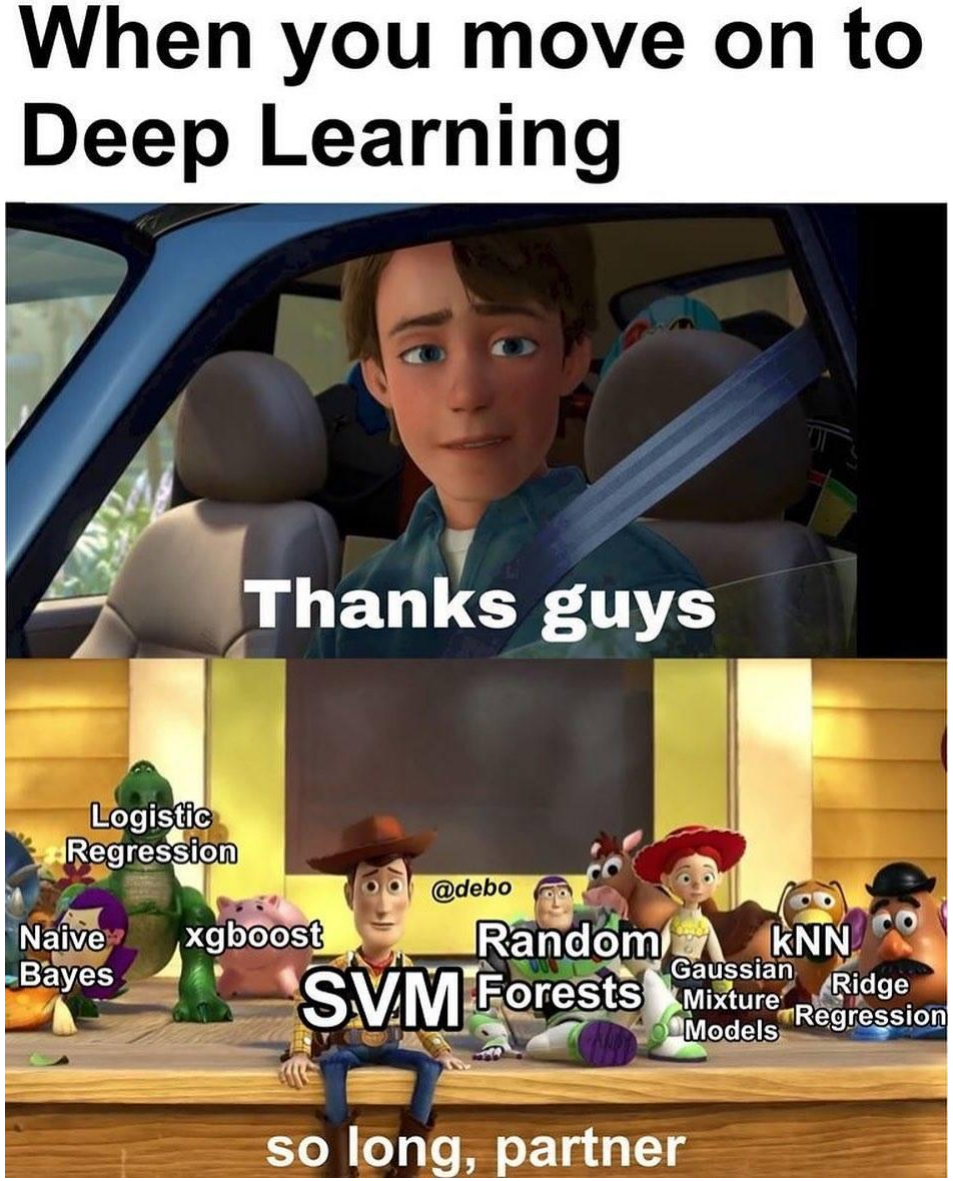
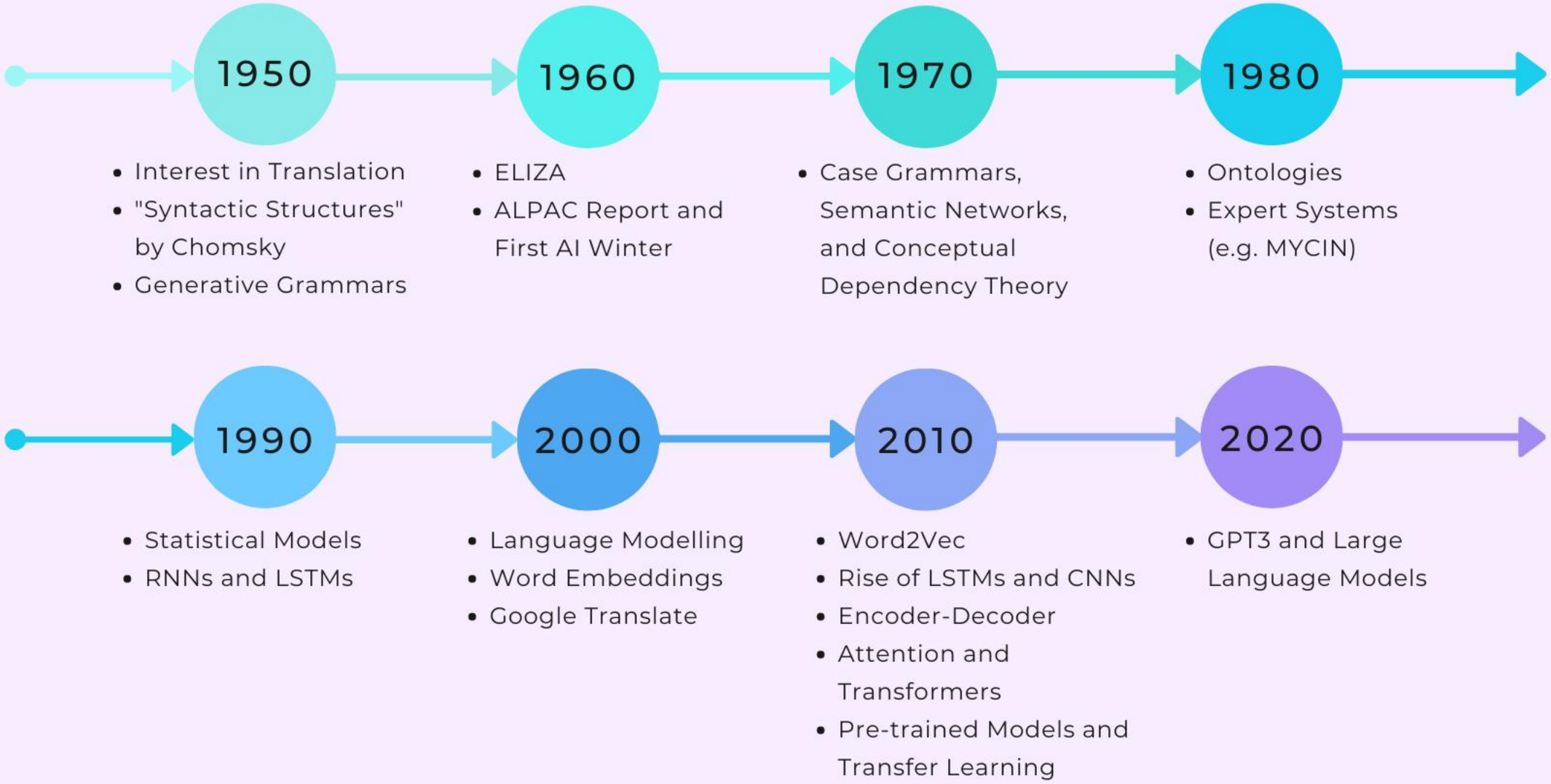


Derin Öğrenme Modelleri



A Brief Timeline of NLP



AN NLP TIMELINE AND THE TRANSFORMER FAMILY

BAG OF WORDS (BOW)
Count the occurrences of each word in the documents and use them as features.
1954

TF-IDF
The BOW scores are modified so that rare words have high scores and common words have low scores.
1972

WORD2VEC
Each word is mapped to a high-dimensional vector called word embedding, which captures its semantic. Word embeddings are learned by a neural network looking for word correlations on a large corpus.
2013

RNN
RNNs compute document embeddings leveraging word context in sentences, which was not possible with word embeddings alone.

LSTM
Capture long term dependencies.
1997

Bidirectional RNN
Capture left-to-right and right-to-left dependencies.
1997

Encoder-decoder RNN
An RNN creates a document embedding (i.e. the encoder) and another RNN decodes it into text (i.e. the decoder).
2014

1986

TRANSFORMER
An encoder-decoder model that leverages attention mechanisms to compute better embeddings and to better align output to input.
2017

BERT
Bidirectional Transformer pretrained using a combination of Masked Language Modeling and Next Sentence Prediction objectives. It uses global attention.
2018

GPT
The first autoregressive model based on the Transformer architecture.

GPT-2
A bigger and optimized version of GPT, pre-trained on WebText.
2019

GPT-3
A bigger and optimized version of GPT-2, pre-trained on Common Crawl.
2020

2018

CTRL
Similar to GPT but with control codes for conditional text generation.
2019

TRANSFORMER-XL
It's an autoregressive Transformer which can reuse previously computed hidden-states to attend to longer context.
2019

ALBERT
A lighter version of BERT, where (1) Next Sentence Prediction is replaced by Sentence Order Prediction, and (2) parameter-reduction techniques are used for lower memory consumption and faster training.
2019

ROBERTA
Better version of BERT, where (1) the Masked Language Modeling objective is dynamic, (2) the Next Sentence Prediction objective is dropped, (3) the BPE tokenizer is employed, and (4) better hyperparameters are used.
2019

XLM
Transformer pre-trained on a corpus of several languages using objectives like Causal Language Modeling, Masked Language Modeling, and Translation Language Modeling.
2019

XLNET
Transformer-XL with a generalized autoregressive pre-training method that enables learning bidirectional dependences.
2019

PEGASUS
A bidirectional encoder and a left-to-right decoder pre-trained with Masked Language Modeling and Gap Sentence Generation objectives.
2019

DISTILBERT
Same as BERT but smaller and faster, while preserving over 95% of BERT's performances. Trained by distillation of the pre-trained BERT model.
2019

XLM-ROBERTA
RoBERTa trained on a multilanguage corpus with the Masked Language Modeling objective.
2019

BART
A bidirectional encoder and a left-to-right decoder trained by corrupting text with an arbitrary noising function and learning a model to reconstruct the original text.
2019

CONVBERT
Better version of BERT, where self-attention blocks are replaced with new ones that leverage convolutions to better model global and local context.
2019

FUNNEL TRANSFORMER
A type of Transformer that gradually compresses the sequence of hidden states to a shorter one and hence reduces the computation cost.
2020

REFORMER
A more efficient Transformer thanks to local-sensitive hashing attention, axial position encoding and other optimizations.
2020

T5
A bidirectional encoder and a left-to-right decoder pre-trained on a mix of unsupervised and supervised tasks.
2020

LONGFORMER
A Transformer model replacing the attention matrices with sparse matrices for higher training efficiency.
2020

PROPHETNET
A Transformer model trained with the Future N-gram Prediction objective and with a novel self-attention mechanism.
2020

ELECTRA
Same as BERT but lighter and better. The model is trained with the Replaced Token Detection objective.
2020

SWITCH TRANSFORMER
A sparsely-activated expert Transformer model that aims to simplify and improve over Mixture of Experts.
2021

NLPLANET The community of NLP enthusiasts!
https://www.linkedin.com/company/nlplanet
https://twitter.com/nlplanet_
https://medium.com/nlplanet

By Fabio Chiusano

Popüler NLP Derin Öğrenme Araştırma Konuları:

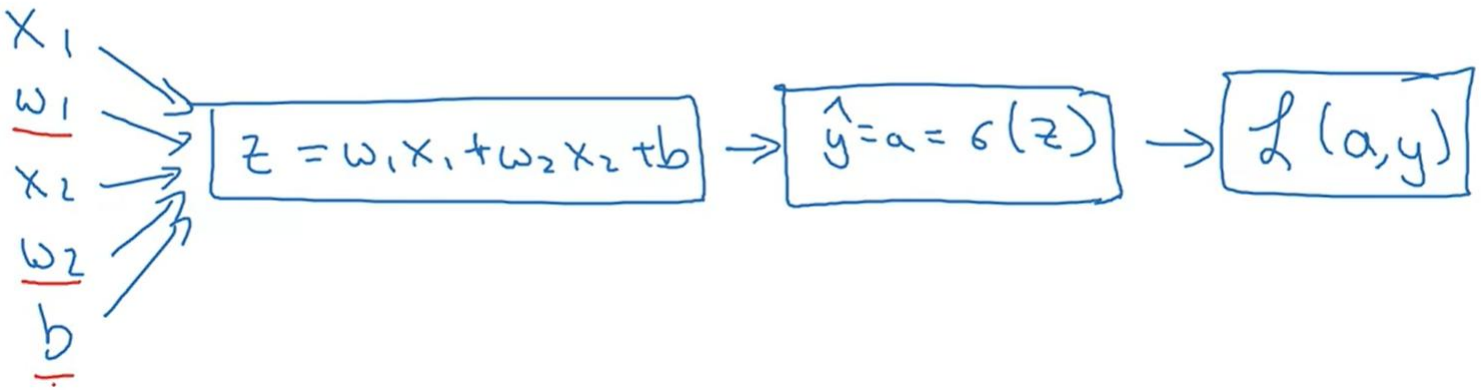
- Dilden Üretim (Text-to-Image, Text-to-Video -> Diffusion: (e.g. CLIP, Imagen, DALL-E 2, Stable Diffusion)
- Dil Üretimi (Seq-to-Seq)
- Makine çevirisi - “Çok dillilik”
- NLP için Derin Öğrenme Modellerinin Yorumlanabilirliği ve Analizi
- Bilgi Çıkarımı (Duygu Analizi, Cümledeki Fikir vb.)
- Büyük Dil Modellerinde Etik
- Diyalog ve İnteraktif Sistemler


NLP Tasks
Text Preprocessing
Coreference Resolution
Part Of Speech Tagging
Word Sense Disambiguation
Grammatical Error Correction
Feature Extraction
Text-to-Data and viceversa
Speech-to-Text
Text-to-Image
Data-to-Text
Text-to-Speech
Fake News and Hate Speech Detection
Fake News Detection
Stance Detection
Hate Speech Detection
Text Reasoning
Common Sense Reasoning
Natural Language Inference
Chatbots
Slot Filling
Intent Detection
Keyword Extraction
Topics and Keywords
Keyword Extraction
Topic Modeling
Knowledge Bases, Entities and Relations
Entity Linking
Named Entity Recognition
Relation Prediciton
Relation Extraction
Text-to-Text Generation
Paraphrase Generation
Lexical Normalization
Text Simplification
Text Summarization
Text Generation
Machine Translation
Information Retrieval and Document Ranking
Question Answering
Sentence/Document Similarity
Classification
Sentiment Analysis
Text Classification
Image by NLPlanet
NLP

Öğrenme Nedir?

Bu basit yapı: Lojistik Regresyon

$$z = w^T x + b$$
$$\hat{y} = a = \sigma(z)$$
$$\mathcal{L}(a, y) = -(y \log(a) + (1 - y) \log(1 - a))$$



z : Hesaplanan Vektör
w : Ağırlık
x : Girdi (input)
b : Bias
L : Hata Fonksiyonu
ŷ : k sayıda sınıfın her biri için çıktı olasılıklarının vektörü
σ : Sigmoid (Softmax)
Sigmoid binary dağılım için, softmax 3 sınıf ve sonrası için kullanılır. Softmax Sigmoid'in genelleştirilmiş versiyonudur.

Sigmoid'in bir takım avantajları var; bir sayıyı alır [0, 1] aralığına eşler ve bu aslında tam da bir olasılık için istediğimiz şeydir. 0 civarında neredeyse lineer olduğundan, ve uçlara doğru düzleştiğinden, aykırı değerleri 0 veya 1'e doğru itme eğilimindedir. Türevlenebilir olması da öğrenme için kullanışlı olmaktadır.

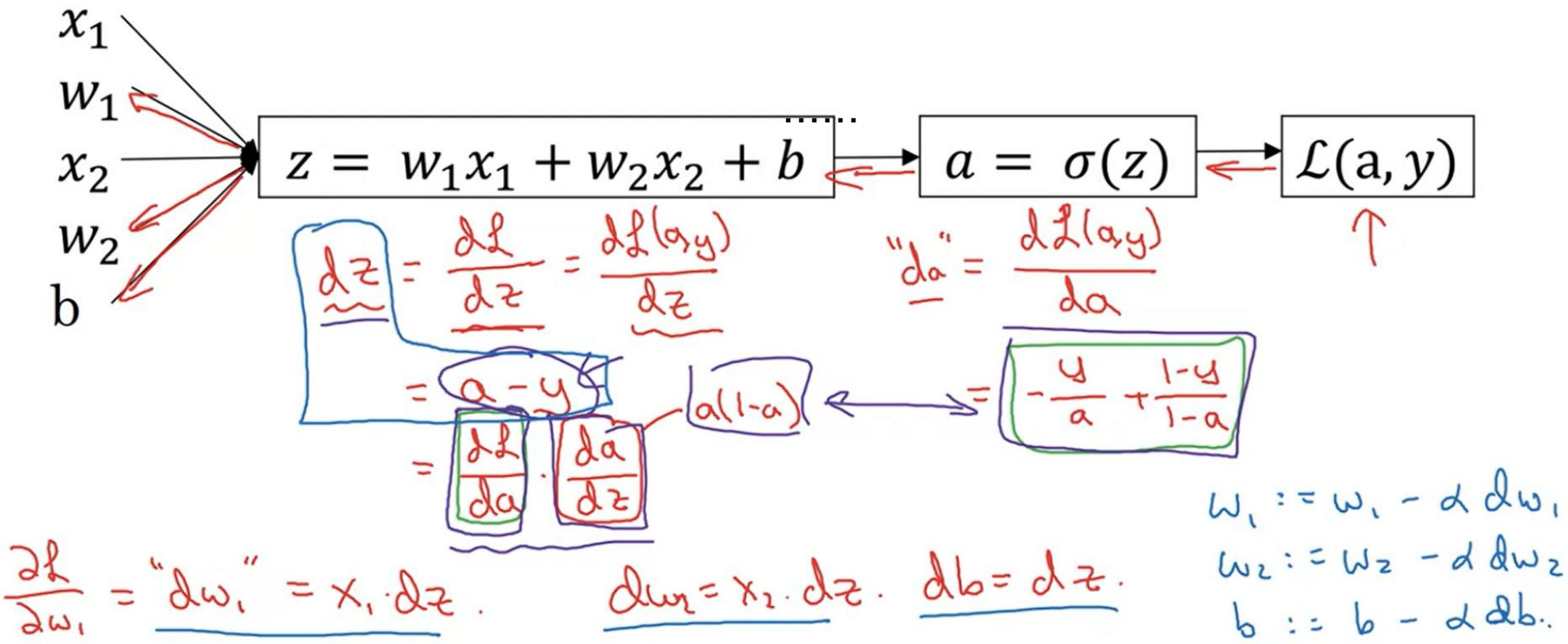
Andrew NG

Backpropagation (Geri besleme) - (1986)

Geri yayılım fikri 1960 - 1970'te ortaya çıktı, ancak sinir ağlarını eğitmek için öğrenme prosedürü olarak resmen kullanılması 1986 yılına kadar ulaştı. Bu, Rumelhart, Hinton ve Williams'ın Nature'da yayınlanan Learning Representations by Back-propagating Errors adlı ünlü makalesinde başarılmıştır.

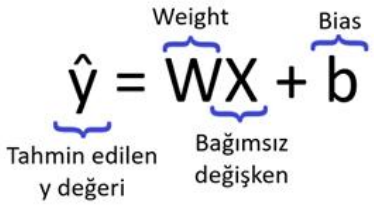
- Özetle, hesaplanan hata bilgisi, türev alarak geriye doğru gönderilir.

Backprop: Eeey weight ve bias, kendinize gelin böyle bir yere varılmaz.



Andrew NG

Terminoloji



Temel Yapay Nöron Eğitiminin Bileşenleri

- x ve **ŷ,** Girdi ve çıktı değerlerimiz. x (bir veri kümesindeki özellikler veya nitelikler olarak düşünülebilir) ile modele sağlanan bilgiler ile bir karar verip sonucu ŷ ile tasvir ediyoruz. Daha sonra bu ŷ değerini gerçek değer ile karşılaştırıp ne kadar hata yaptığımızı anlıyoruz.
- Weight (Ağırlık), iki nöron arasındaki sinyali (veya bağlantının gücünü) kontrol eder. Başka bir deyişle, bir ağırlık, girdinin çıktı üzerinde ne kadar etkiye sahip olacağına karar verir.
- Bias, girdilerin ve ağırlıkların çarpımına eklenen sabit bir değerdir (veya sabit bir vektördür)
  - sonucu dengelemek için kullanılır
  - aktivasyon fonksiyonunun sonucunu pozitif veya negatif tarafa kaydırmak için kullanılır
- Ağırlıklar ve Bias (genellikle w ve b olarak adlandırılır), sinir ağları dahil olmak üzere bazı makine öğrenimi modellerinin öğrenilebilir parametreleridir.
- Aktivasyon Fonksiyonu, bir yapay sinir ağındaki her düğüm, ağırlıklı girdileri ve biaslarının toplamı üzerinde matematiksel bir işlem gerçekleştirir ve bir çıktı üretir. (Fonksiyon, her girdinin tek bir çıktıya sahip olduğu özel bir ilişkidir. Ör: Sigmoid)
- Hata (Maliyet) Fonksiyonu (Loss Function), modelin doğruluğunu gösterir. Çıktısı, sinir ağına, modelin doğruluğunu artırmak için ağırlıkların ve bias'ların ayarlanması gerekip gerekmediğini söyler. Maliyet fonksiyonunu, makineyi başarı için ödüllendirme ve/veya başarısızlık için cezalandırma aracı olarak düşünebiliriz. Makinenin başarılarından veya hatalarından ders çıkarmasını sağlar.

Overfitting

- Bir modelin x girdilerine karşılık gelen y çıktıları ile birlikte eğitildiğini kabul edelim. (x,y) ilişkisininin bir tahmini fonksiyonunu (Y) elde etmeye çalışalım.
- Bu fonksiyonun veri setine çok fazla uyması, veri setini ezberlemesi demektir (overfitting).
- Overfitting, her veriye tamamen uyacak şekilde bir fonksiyonun geliştirilmesi ve karmaşık fonksiyonlar üretebilmesi durumlarında ortaya çıkar.
- Modelden daha önce görmediği bir veriyi tahmin etmesi istendiğinde veri setini ezberlediğinden doğru tahmin yapamaz.

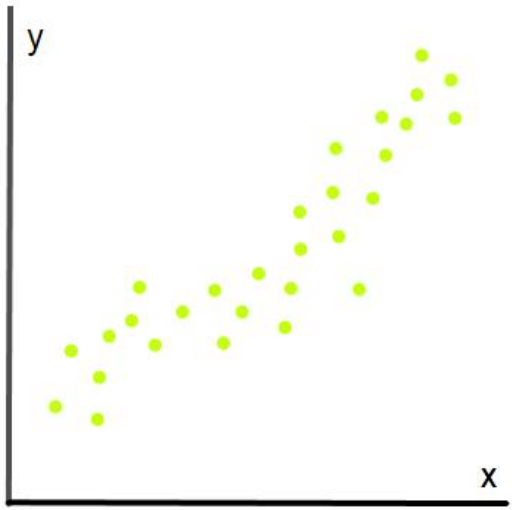
Örnek: Bir öğrenci sınava hazırlanırken konulara çalışmak yerine sadece önceki çıkmış soruları ezberler ve sınavda da daha önce hiç görmediği sorular geldiğinde başarısız olur.

Underfitting

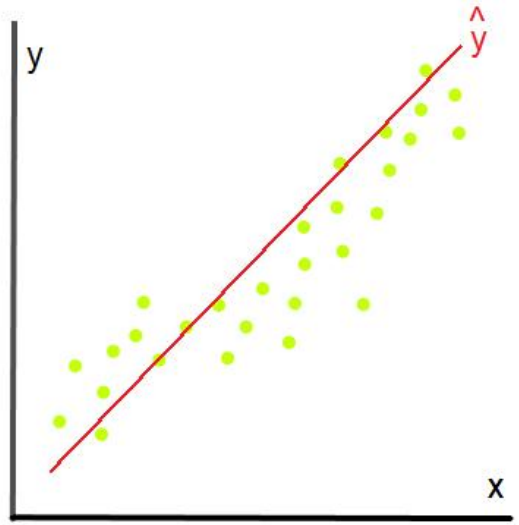
- Y tahmini fonksiyonunun veri setinden çok bağımsız olması yani veri setinin ilişkisini yansıtamaması durumudur (underfitting).
- Underfitting, modelin çok basit olması yani verinin yalnızca çok küçük bir kısmını ifade edebilecek yeterlilikte fonksiyon üretmesidir.

Örnek: Öğrenci eğer konuların sadece bir kısmını çalışırsa ve sınavda tüm konulardan sorumlu olacağı için başarısız.

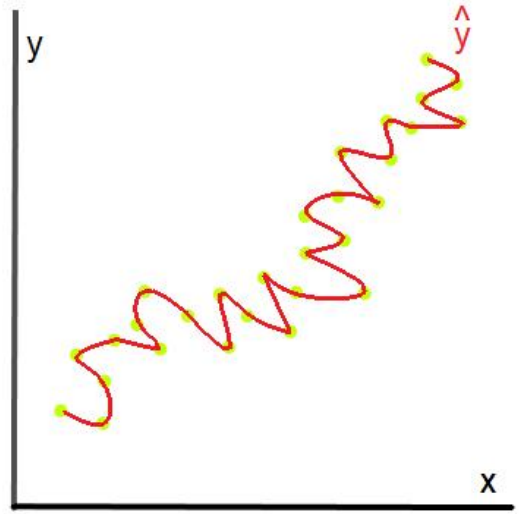
Modelden beklentimiz veri setini ezberlememesi
ama aynı zamanda veri setinin geneline de hakim olmasıdır.



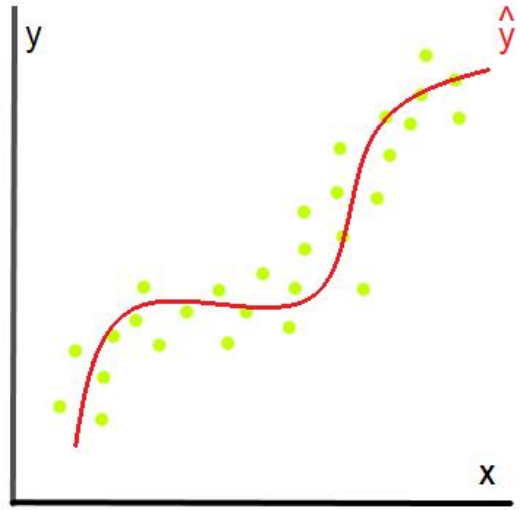
Veri Seti



Underfitting



Overfitting



İdeal Model

Hiperparametre

Veri setini hazırlarken, modeli oluştururken veya eğitirken kullanıcı tarafından belirlenmesi gereken bileşenler.

- KNN algoritmasında k nın değeri ?
- SVM algoritmasında kernel fonksiyonu?
- Derin öğrenme modellerindeki learning rate, mini batch, epoch, gizli katman sayısı, kaç node içereceği, başlangıç ağırlıkları, aktivasyon fonksiyonu...

Ne olması gerektiği, modeli tasarlayan kişiye bırakılmış, probleme, veri setine göre değişiklik gösteren parametrelere hiper-parametre denir.

- Modelin yüksek başarım sağladığı birbirinden farklı hiper parametre grupları olabilir.
- En uygun hiper parametre grubunun seçilmesi önemli bir problemdir.
- Tasarımcının sezgisine, önceki problemlerden elde edilen tecrübelere, farklı alanlardaki uygulamaların kendi problemimize yansımasına, güncel trendlere, vb. göre değişebilir.
- En uygun hiper parametre grubunun en optimum şekilde seçilmesine yönelik farklı teknikler (stochastic gradient descent, adagrad, adadelta, adam, adamax) kullanılır.

Derin Öğrenmede En Sık Kullanılan Hiperparametreler

Veri Setinin Boyutu

- Derin öğrenme uygulamalarında veri setinin büyüklüğü ve çeşitliliği öğrenme için önemlidir. Veri seti ne kadar büyük ise öğrenme o kadar iyi olur (“devasa boyutlarda veri kullanmadan aynı performansa nasıl ulaşırız/yaklaşırız?” sorusu da ayrı bir araştırma konusudur: ör. distillation modeller (DistilBERT).
- Veri setinin büyüklüğü öğrenme için harcanan zamanı ve modelin büyüklüğünü de arttırır.
- Eğitimin sık yapılmayacağı ve depolama alanının sorun olmadığı uygulamalarda öğrenim başarısı tercih edilir.
- Eğitimin sık yapılacağı veya mobil ortamlar gibi depolama alanının problem olacağı uygulamalarda başarımın yanı sıra veri setinin büyüklüğü de değerlendirilmelidir.
- Veri setinin büyük olması her zaman iyi bir model için yeterli değildir, veri setindeki çeşitlilik de önemlidir.

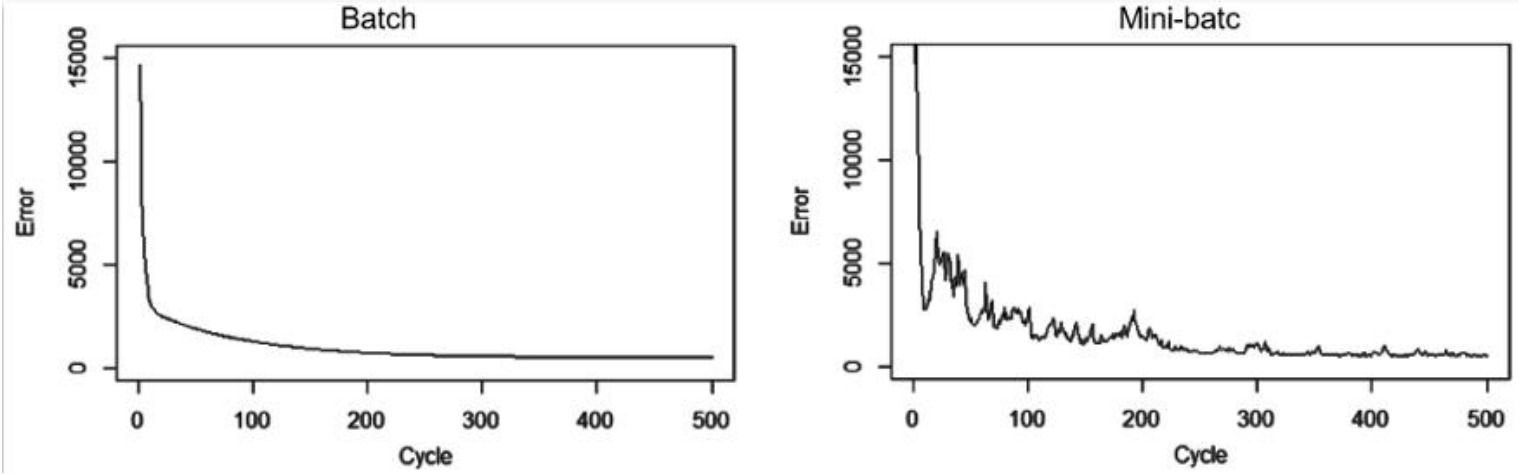
- Veri seti büyüdükçe başarım sonsuza kadar doğru orantılı olarak artmaz, belli bir noktadan sonra artışlar küçük olur ya da daha da kötüleştirebilir (kullandığımız model bu kadar bilgi girişinin altından kalkabilecek kadar iyi olmayabilir).
- Modeli eğitirken veri setinin hangi noktada kırıldığına da dikkat edilmelidir.
- Transfer Learning ile öznitelik transferi yapılabilir. Veri seti küçük ise sentetik veri ile artırılabilir (her zaman her konuda yeterince kaynak veya paylaşım olmayabiliyor).
- Görsel veriler üzerinde çalışılıyorsa her sınıf için binlerce örnek olmalıdır.
- Sınıflar birbirine çok benziyorsa, veri seti içerisinde geçişgenlik varsa «batch» değeri değiştirilir.

“Mini-Batch” Boyutu

- Modele eğitim sürecinde tüm veri setini girdi olarak vermek yerine, veriyi küçük parçalar halinde bölmemizi sağlayan hiperparametredir.
- Parametreler eğitim esnasında güncellenerek en optimal sonuca ulaşırken, güncelleme işlemi tüm veri seti işlendikten sonra gerçekleştirilirse hem zaman kaybına hem de işlem maliyetinin artmasına neden olur.
- Öğrenmenin her iterasyonunda geriye yayılım (“backpropagation”) işlemi ile ağ üzerinde geriye dönük olarak gradyan (“gradient descent”) hesaplaması yapılarak ağırlık değerleri güncellenir.
- Bu hesaplama işleminde veri sayısı ne kadar fazla ise hesaplama da o kadar uzun sürer.
- Bu yüzden veri seti parçalara ayrılarak, güncelleme işleminin yapılması sağlanır (parçaların büyüklüğü de batch_size olarak adlandırılır).
- Batch_size belirlenirken veri setinin büyüklüğü, verinin dağılımı ve makinenin işlem gücü dikkate alınmalıdır.
- batch_size=1 seçilirse, her veri sonrası güncelleme yapılır (schoastic gredient decent). Sırasıyla gelen veriler birbirlerinden çok farklı ise tahmin sonuçlarımız da o kadar farklı olur. Bu istenmeyen bir durumdur. batch_size= toplam veri sayısı seçilirse (batch gredient descent), güncelleme sayısı artacağından makinenin işlem gücü yetersiz kalır (batch_size a değer atanmazsa otomatik olarak batch_size=toplam veri sayısı olarak alınır).

stochastic gredient decent mini-batch=1	batch gredient descent mini-batch=tüm veri seti
▪ Modelin gürültüyü öğrenmesine neden olabilir. Grafikteki zikzaklar artar. Öğrenme sürecinde her defasında farklı veri kullanılır. ▪ Local optimum'da takılıp, global optimuma hiç ulaşamayabilir. ▪ Salınımı azaltacak yöntemler kullanılır RMSprob (Root Mean Square error Propability).	▪ Model daha az gürültü öğrenecektir. ▪ Bütün veriyi işlediği için öğrenme çok uzun sürecektir. ▪ Optimizasyon işleminde büyük adımlarla ilerleme olacaktır.

Batch seçiminde verilerin rastgele seçilmesi önemlidir.



BGD(sol) - Stochastic ya da Mini-batch (sağ) uygulandığı durumda hata değeri değişimi (Resim kaynak : Steven Schmatz)

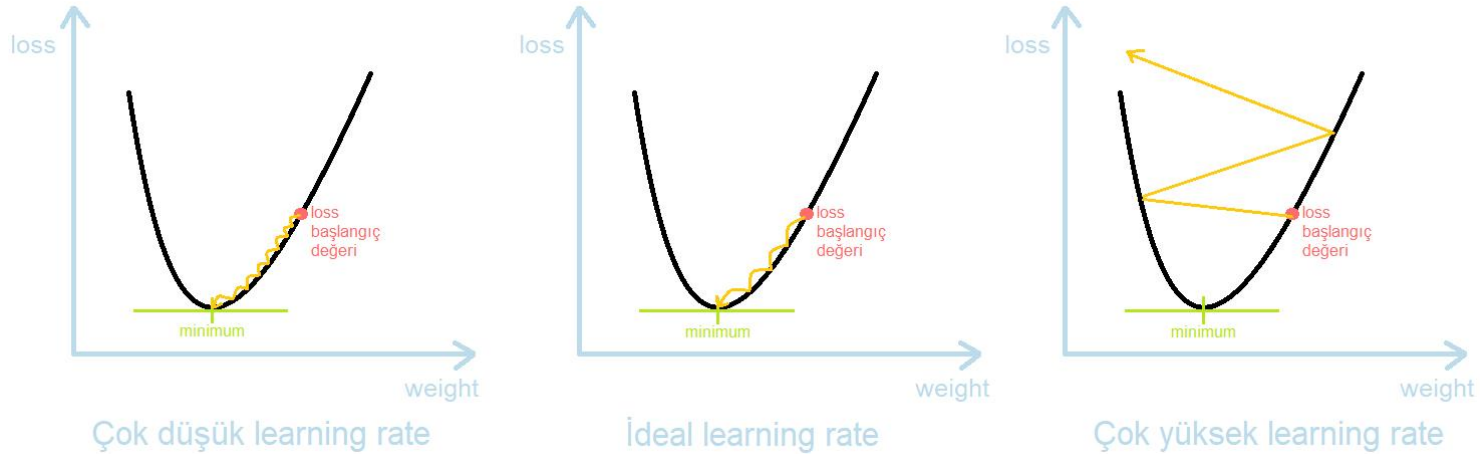
Epoch-Iterasyon

- Tüm verilerin ağ tarafından bir kere işlenmesine 1 epoch denir.
- Verileri tek seferde işlemeyip, mini-batch'lere böldüğümüzde, her bir mini-batch 1 iterasyona karşılık gelir.

20 kıtadan oluşan bir şiir olsun (20 kıta = 1 epoch)
ve ezberlerken 2 kıtaya bölüp ezberleyelim (2 kıta = mini-batch) böylece (iterasyon sayısı=10) olur.

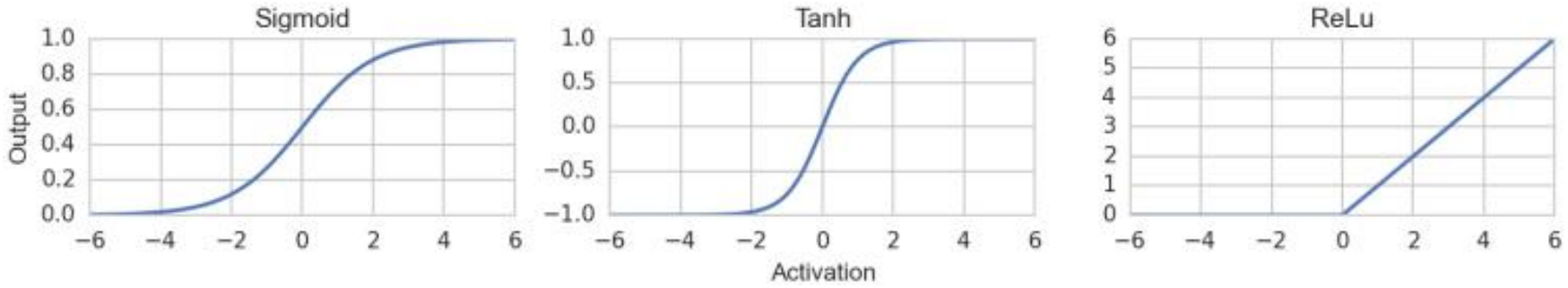
Learning Rate

- Kayıp fonksiyonu üzerinde optimum noktaya (kayıp değerinin minimum olduğu) ulaşmak için gradyan iniş kullanılır. Bu iniş esnasında izlenen yolda atılacak adım büyüklüğü learning rate olarak tanımlanır.
- Learning rate, eğitimden önce belirlenen bir hiperparametredir.
- Çok küçük seçilirse eğitim için gerekli epoch sayısı artar ve eğitimin süresi uzar.
- Çok büyük seçilmesi minimum noktanın atlanmasına ve optimuma ulaşılamayacağı anlamına gelir.
- Bu gibi durumların önlenmesi için ideal bir learning rate değeri seçilmelidir.
- Learning rate için genellikle varsayılan değerler olarak $10^{-1, -2 \text{ ve } -3...}$ ya da $e^{-2, -3, -4...}$ kullanılır. Bu parametrenin optimizasyonu için farklı metodlar bulunmaktadır (cyclical lr, rate reduction vb.).



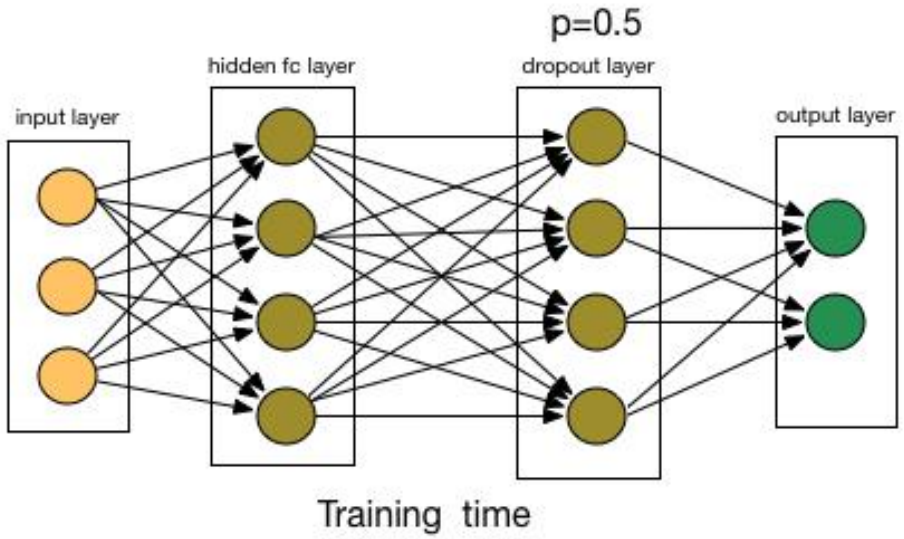
Aktivasyon Fonksiyonu

- Derin öğrenme yöntemleri doğrusal olmayan (non-linear) yapıya sahip problemlerin çözümünde kullanılır.
- Gizli katmanlarda y = f(x,w) şeklindeki lineer fonksiyonumuzda matris çarpımı yapılıp, nöronların ağırlığı hesaplandıktan sonra, çıktı doğrusal olmayan (non-linear) bir değere dönüştürülür.
- Sonucun non-linear hale dönüştürülmesini aktivasyon fonksiyonları sağlar.
- Gizli katmanlarda geri türev alınabilmesi (gradient decent hesaplanabilmesi) için (öğrenmede fark, geri türevle alınır) gizli katmanların çıktısı aktivasyon fonksiyonları ile normalize edilir [0,1].
- Aktivasyon fonksiyonlarının bazıları sigmoid, ReLU, PreLU, …
- ReLU'da parametreler Sigmoid'e göre daha hızlı öğrenir.
- PReLU ise, ReLU'nun kaçırdığı negatif değerleri yakalar (bizim için negatif değerler önemliyse PReLU tercih edilmelidir).



Dropout (Seyreltme) Değeri

- Tam bağlı katmanlarda belli eşik değerin altındaki düğümlerin çıkarılması başarımı arttırabilir.
- Zayıf bilgilerin unutulması, öğrenmeyi hızlandırabilir.



- Seyreltme (Dropout) eşik değeri olarak [0,1] aralığı ve genelde de 0.1-0.5 aralığında kullanılmaktadır.
- Probleme ve veri setine göre değişiklik gösterebilir.
- Seyreltme (Dropout) için rastgele eleme yöntemi de kullanılabilir.
- Tüm katmanlarda aynı dropout değerini kullanmak zorunlu değildir.

Train (Eğitim) Veri Seti

- Modeli eğitmek için kullandığımız veri setidir.

Validation (Doğrulama) Veri Seti

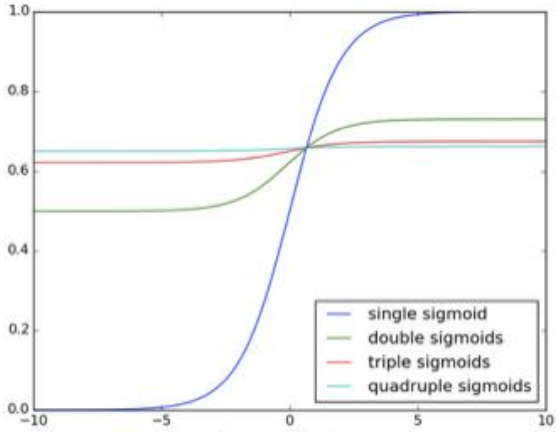
- Modelin hiperparametreleri ayarlanırken eğitim veri kümesine uyan bir modelin tarafsız olarak değerlendirmesini yapmak için kullanılan veri kümesidir.
- Sistemin uygun parametrelerini seçmek için kullanılır. Doğrulama seti, eğitim setinin bir parçası olarak kabul edilebilir çünkü modelinizi, sinir ağlarını veya diğerlerini oluşturmak için kullanılır.
- Modeli test öncesinde eğitirken, öğrenme sürecini bu veri seti ile takip edebiliyoruz. Bu set sayesinde hangi modelin/parametrelerin elimizdeki veri tipine uygun çalıştığını gözlemleyebiliyoruz.

Test Veri Seti

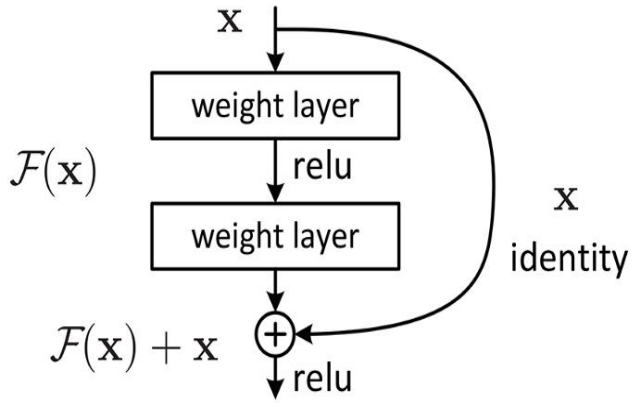
- Model daha önce hiç görmediği bir veri seti ile test edilir.

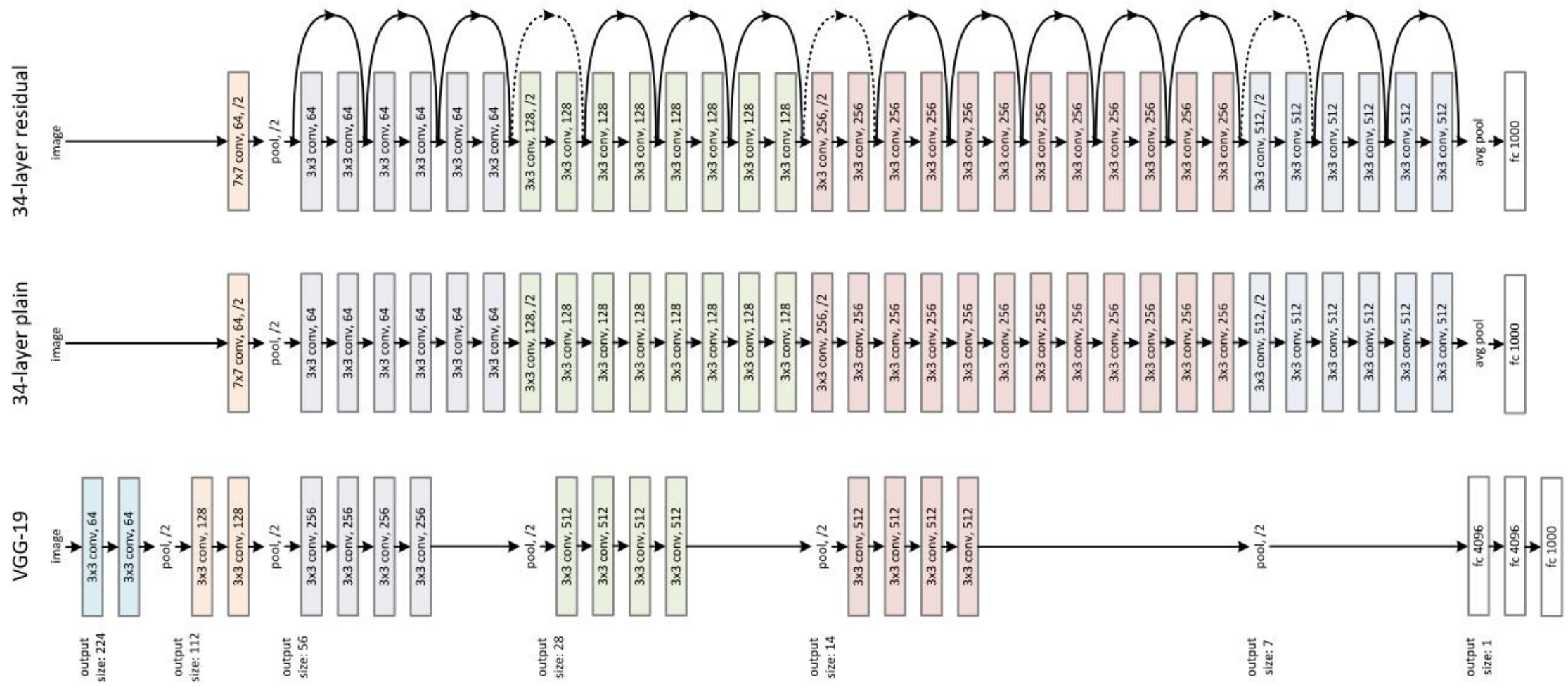
Vanishing/Exploding Gradient (Gradient yok olması/büyümesi)

- Gradient, tüm ağırlıkları ayarlamamızı sağlayan bir değerdir.
- Birbirine bağlı uzun ağlarda hatanın etkisi gittikçe küçülür ve gradient kaybolmaya başlayabilir.
- Bütün katmanlar ve zamana bağlı adımlar birbirine çarpımla bağlı olduğundan, türevleri yok olma veya büyümeye meyillidir.
- Gradient yok olması/büyümesi ağın büyük değerler üretmesini sağlar ve bizi doğru sonuçtan uzaklaştırır. Eşik değer (Threshold) koyarak çok yüksek değerli gradientler atılır.
- Gradientler aşırı küçülerek yok olması durumunu tespit etmek zordur. Ne zaman durdurulması gerektiğini tahmin etmek zordur. Bazı çözümler mevcuttur.
  - W için uygun başlangıç değerleri seçmek
  - Sigmoid ve Tanh aktivasyon fonksiyonları yerine ReLU kullanmak
    ReLU fonksiyonunun türevi 0 veya 1'dir
  - Bu problemi RNN'lerde çözmek için LSTM dizayn edilmiştir. Ancak bu da sadece biraz iyileşme sağlamıştır. Güncel olarak dil konusunda bu probleme en iyi çözüm residual yapıları kullanan Transformers modelidir.



Residual Networks ile vanishing problemi nasıl çözüldü?



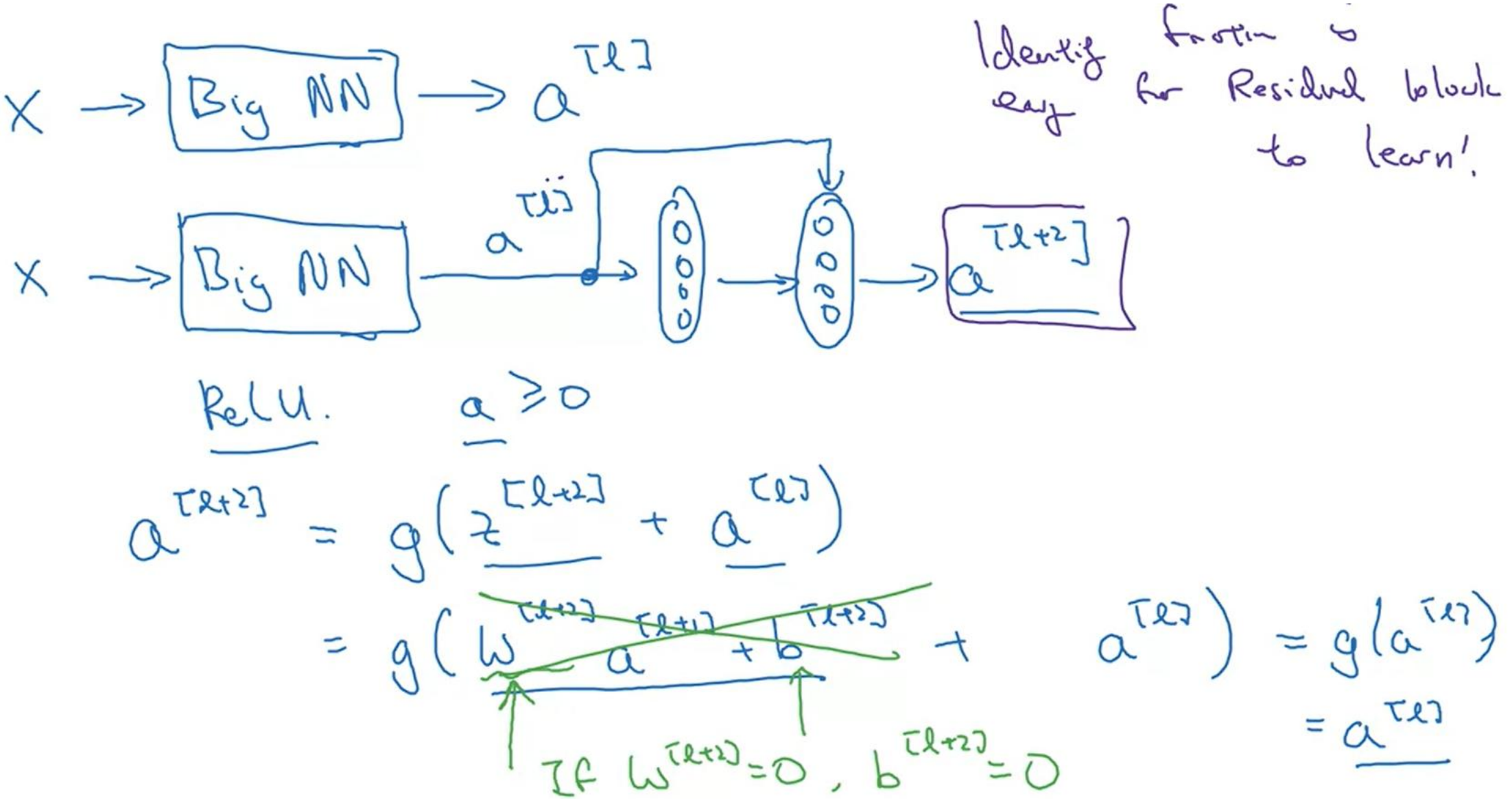


Residual Network'ün kısaltması olan ResNet, 2015 yılında Kaiming He, Xiangyu Zhang, Shaoqing Ren ve Jian Sun tarafından "Deep Residual Learning for Image Recognition" adlı makalelerinde tanıtılan belirli bir sinir ağı türüdür. ResNet, VGG benzeri modeller üzerinde yenilikçi bir yaklaşımdı:

* ILSVRC 2015 sınıflandırma yarışmasında %3,57'lik top-5 hata oranıyla 1. oldu (An Ensemble Model)
* ILSVRC ve COCO 2015 yarışmasında ImageNet Detection, ImageNet yerelleştirme, COCO algılama ve COCO segmentasyonda 1. oldu.
* Faster R-CNN'de VGG-16 katmanlarının ResNet-101 ile değiştirilmesi. %28’lik bir iyileşme gözlemlediler.
* 100 katman ve 1000 katman ile verimli şekilde eğitilmiş ağlar tasarladılar.

Farklı bir domain olmasına rağmen dilde de nasıl aşılması gereken bazı problemlerin çözümünün önünü açtığını Transformers (2017) tasarımında göreceğiz.

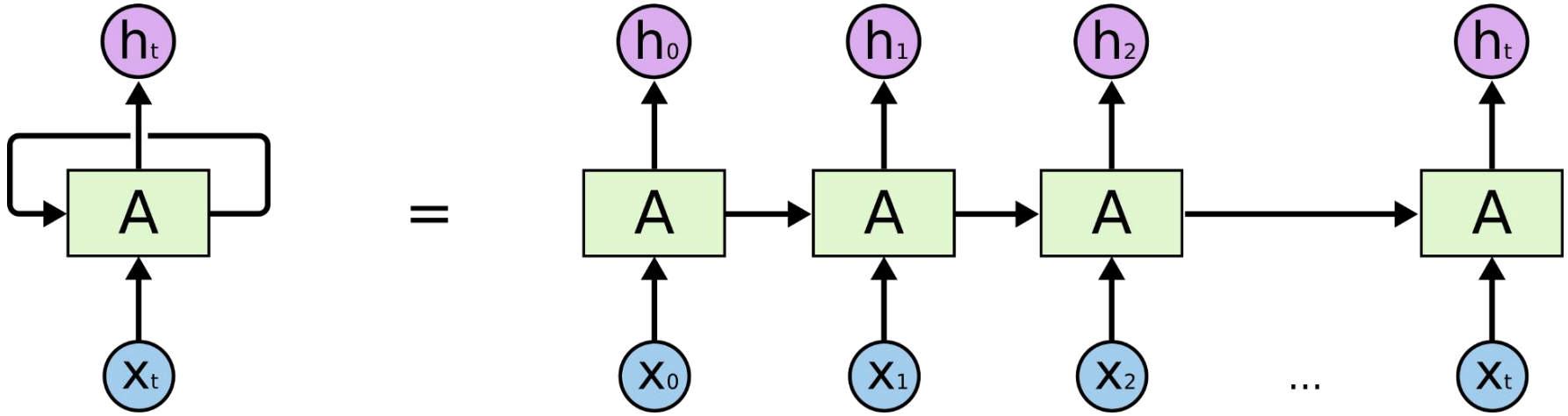
Neden Residual Network işe yarıyor?



Andrew Ng

RNN (Recurent Neural Network-Yinelenen Sinir Ağları) Nedir?

- En büyük farkları zaman bilgisini eğitime dahil etmeleridir. Bellek tutarlar da denilebilir.
- Diğer sinir ağlarında her girdi birbirinden bağımsız iken, RNN'lerde girdiler birbirleri ile ilişkilidir.
- RNN'ler bir sonraki adımı takip edebilmek için girdiler arasında ilişki kurarlar ve eğitilirken tüm bu ilişkileri hatırlarlar (bir noktadan itibaren unutmaya başlarlar).



Kaynak:https://medium.com/@vidishajitani25/recurrent-neural-network-maths-69214e4d69e1

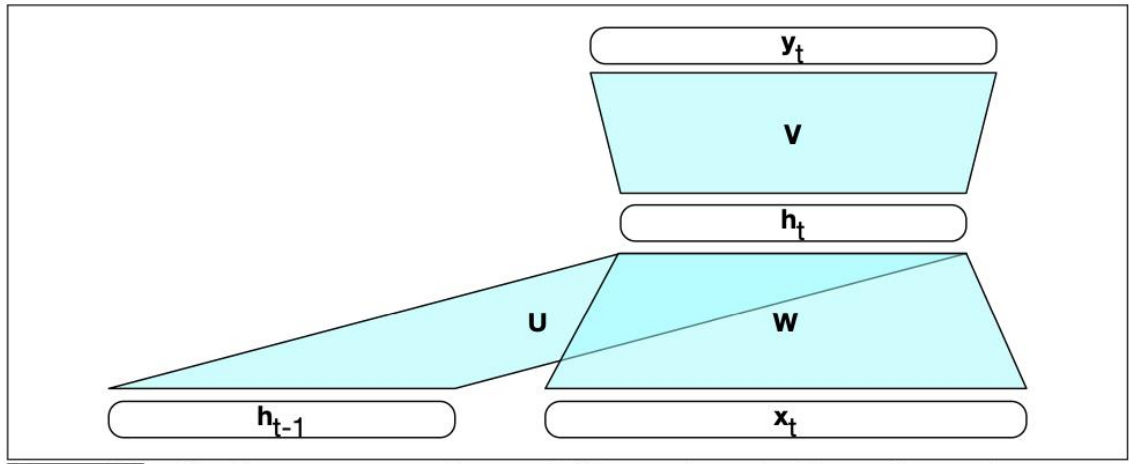


Figure 9.3 Simple recurrent neural network illustrated as a feedforward network.

$$\mathbf{h}_t = g(\mathbf{U}\mathbf{h}_{t-1} + \mathbf{W}\mathbf{x}_t)$$
$$\mathbf{y}_t = f(\mathbf{V}\mathbf{h}_t)$$

$$W \in \mathbb{R}^{d_h \times d_{in}}$$
$$\mathbf{U} \in \mathbb{R}^{d_h \times d_h}$$
$$\mathbf{V} \in \mathbb{R}^{d_{out} \times d_h}$$

h_t : Şu anki h değeri
h_{t-1} : Bir önceki h değeri
x_t : Şu anki girdi vektörü

W : Input Ağırlık Matrisi
U : Önceki katmanların Ağırlık Matrisi
V : Çıktı katmanı Ağırlık Matrisi

- RNN ile gelen en önemli değişiklik, gizli katmanı önceki zaman adımından mevcut gizli katmana bağlayan yeni ağırlıklar U'da bulunmaktadır. Bu ağırlıklar, ağın mevcut girdi için çıktıyı hesaplarken geçmiş bağlamı nasıl kullandığını belirler. Ağdaki diğer ağırlıklarda olduğu gibi, bu bağlantılar da geri yayılım yoluyla eğitilir.
- Bir ağa, geçmiş bilgileri artımlı şekilde dahil etmenin sebebi, belli bir düzende gelen girdi verisinin, çıktı için bir anlamı olmasıdır. Bu çeşit girdi verileri için geleneksel ağlar yeterli olmaz.
- Yazı, konuşma ve zamana bağlı çeşitli sensörlerden belli bir sıra ile gelen verilerin yapısını anlamada kullanılırlar.

RNN Eğitimi

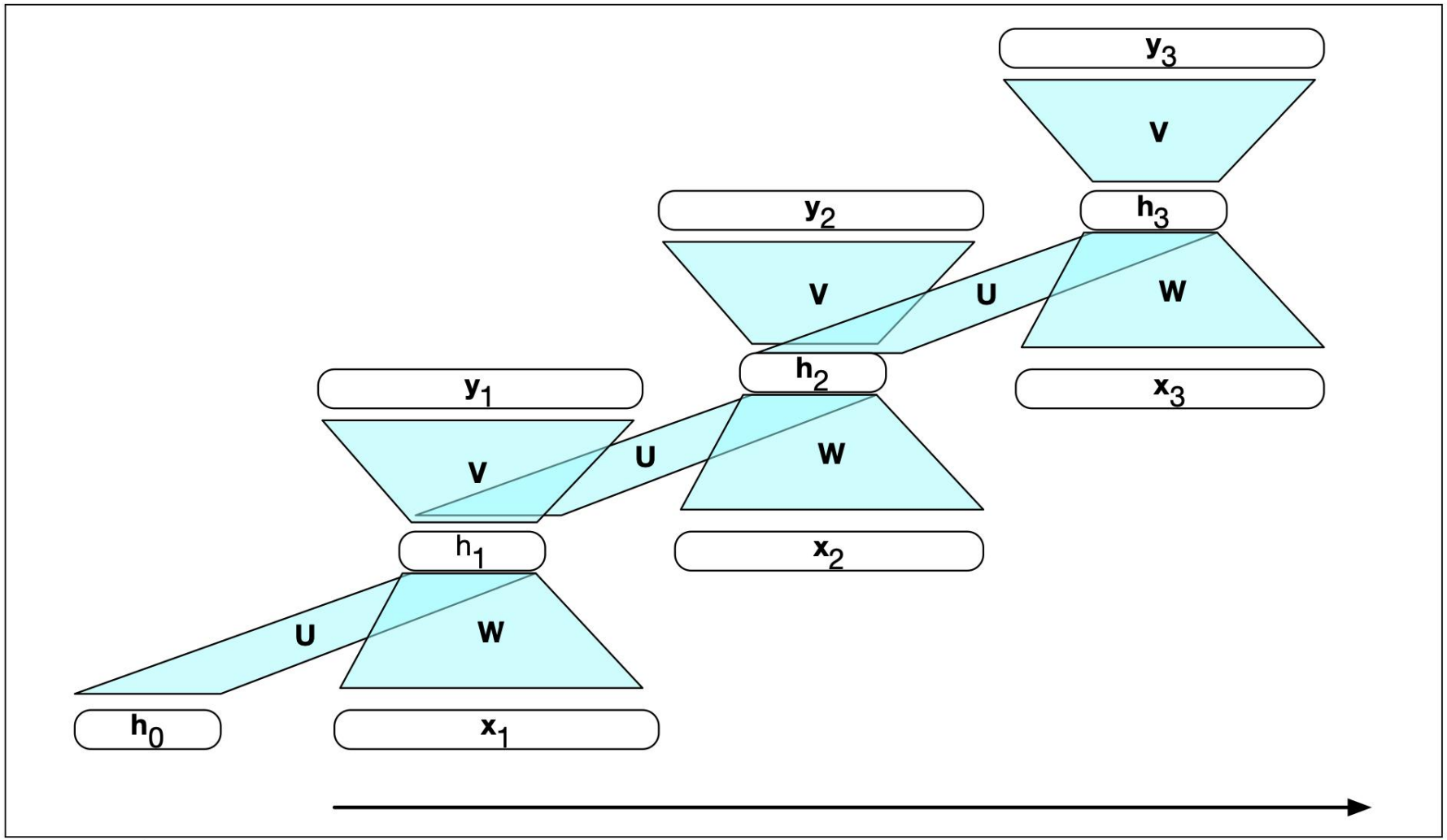


Figure 9.5 A simple recurrent neural network shown unrolled in time. Network layers are recalculated for each time step, while the weights **U**, **V** and **W** are shared in common across all time steps.

Dil Modelleri Olarak RNN'ler

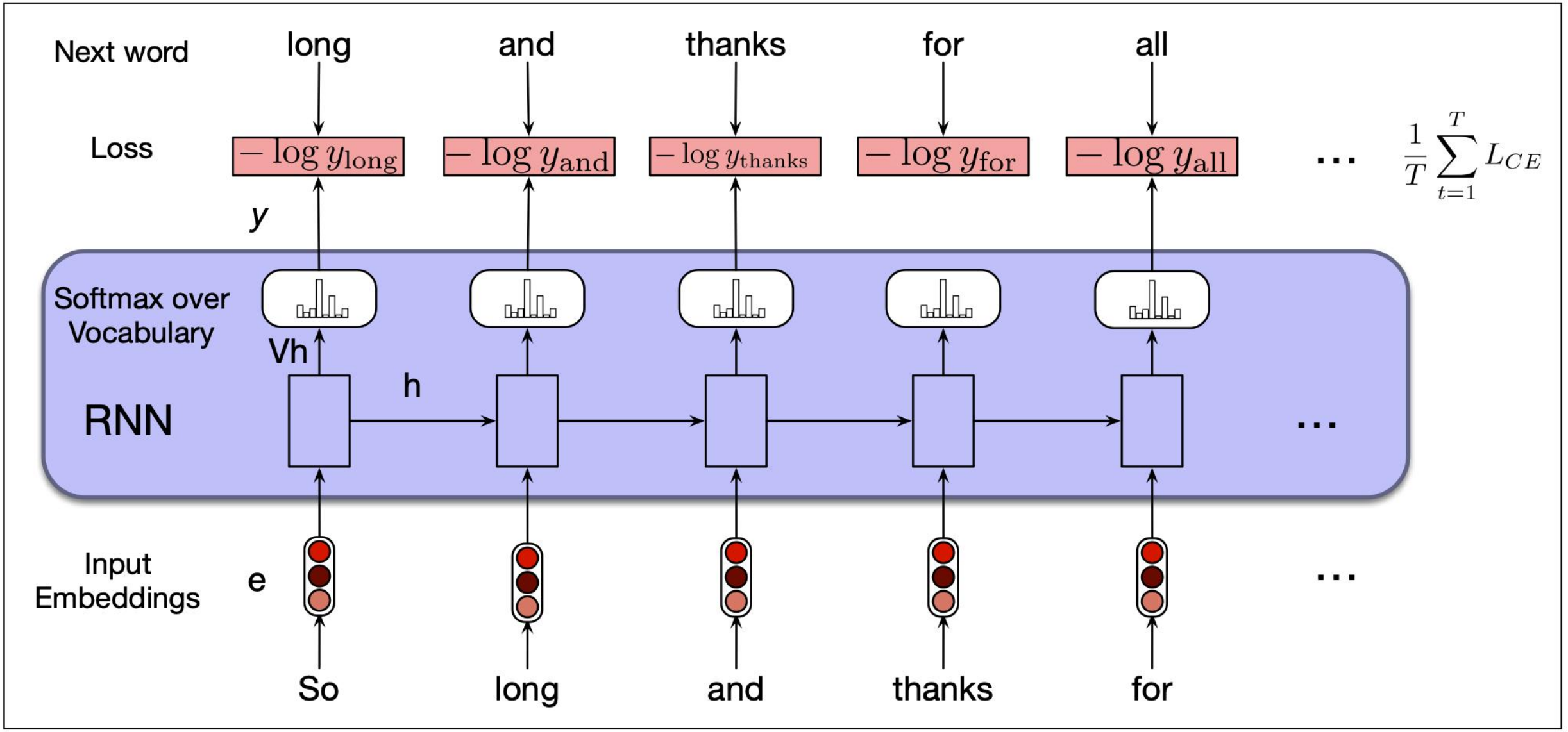


Figure 9.6 Training RNNs as language models.

Dizi etiketleme

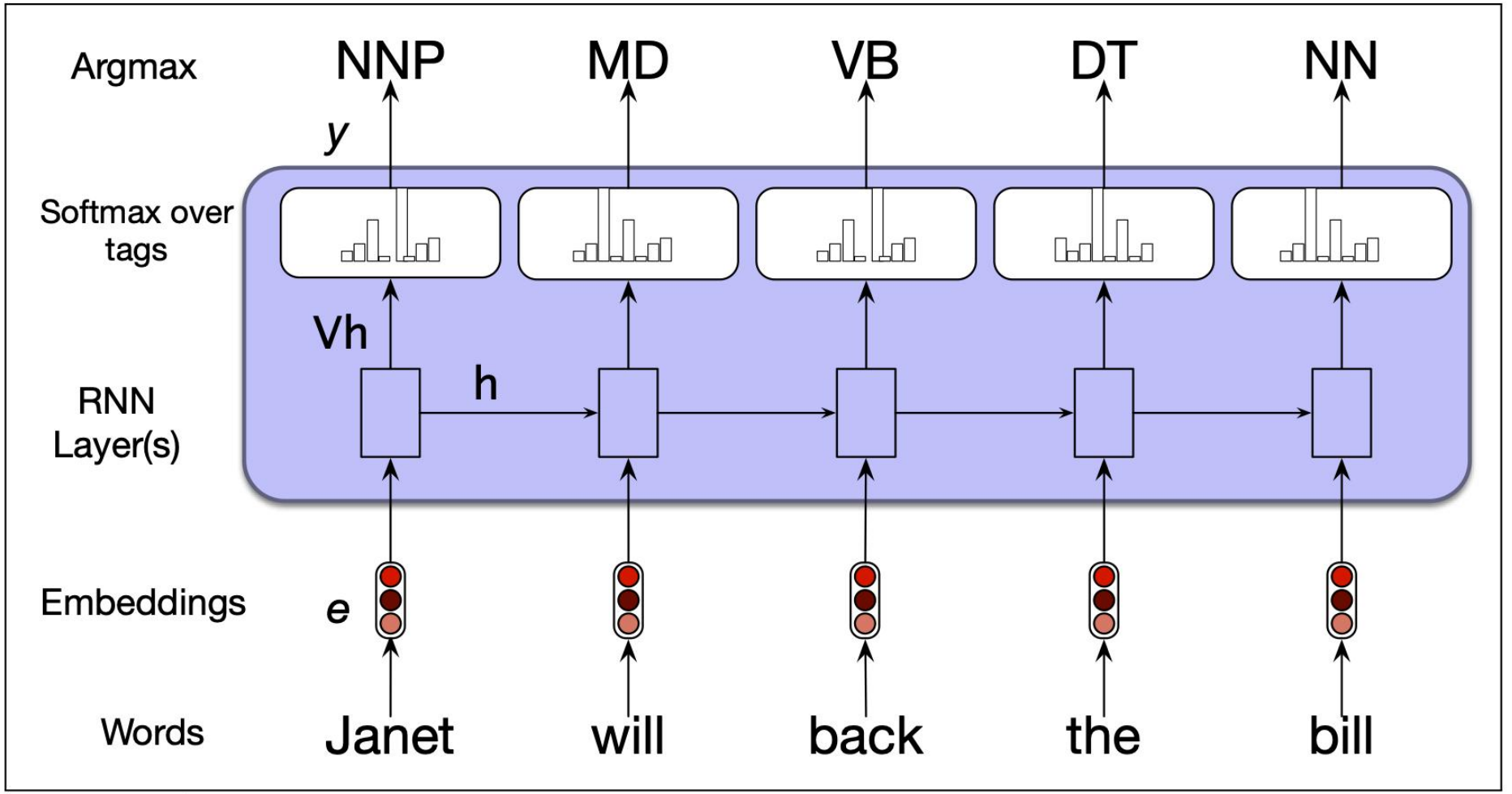


Figure 9.7 Part-of-speech tagging as sequence labeling with a simple RNN. Pre-trained word embeddings serve as inputs and a softmax layer provides a probability distribution over the part-of-speech tags as output at each time step.

Sınıflandırma

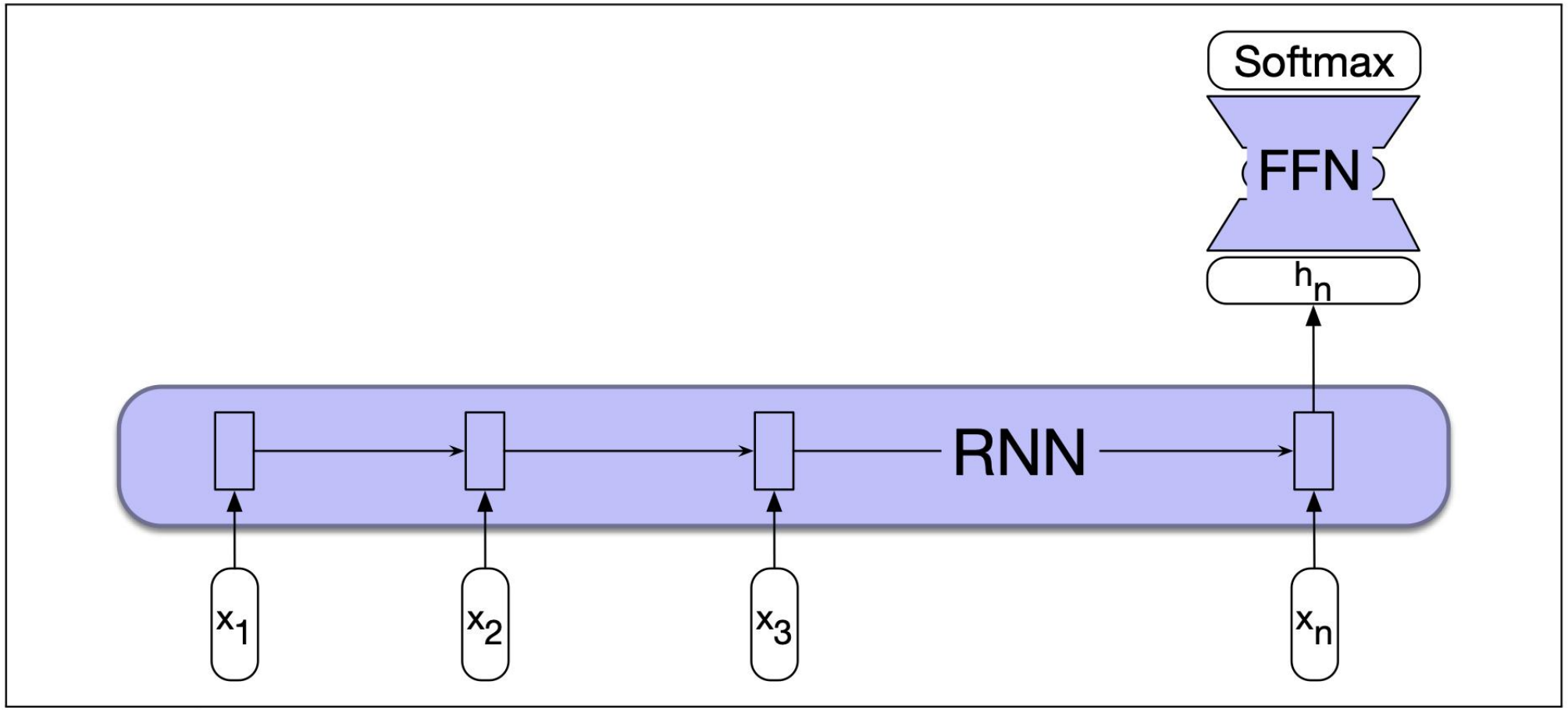


Figure 9.8 Sequence classification using a simple RNN combined with a feedforward network. The final hidden state from the RNN is used as the input to a feedforward network that performs the classification.

RNN Tabanlı Dil Modelleri ile Üretim

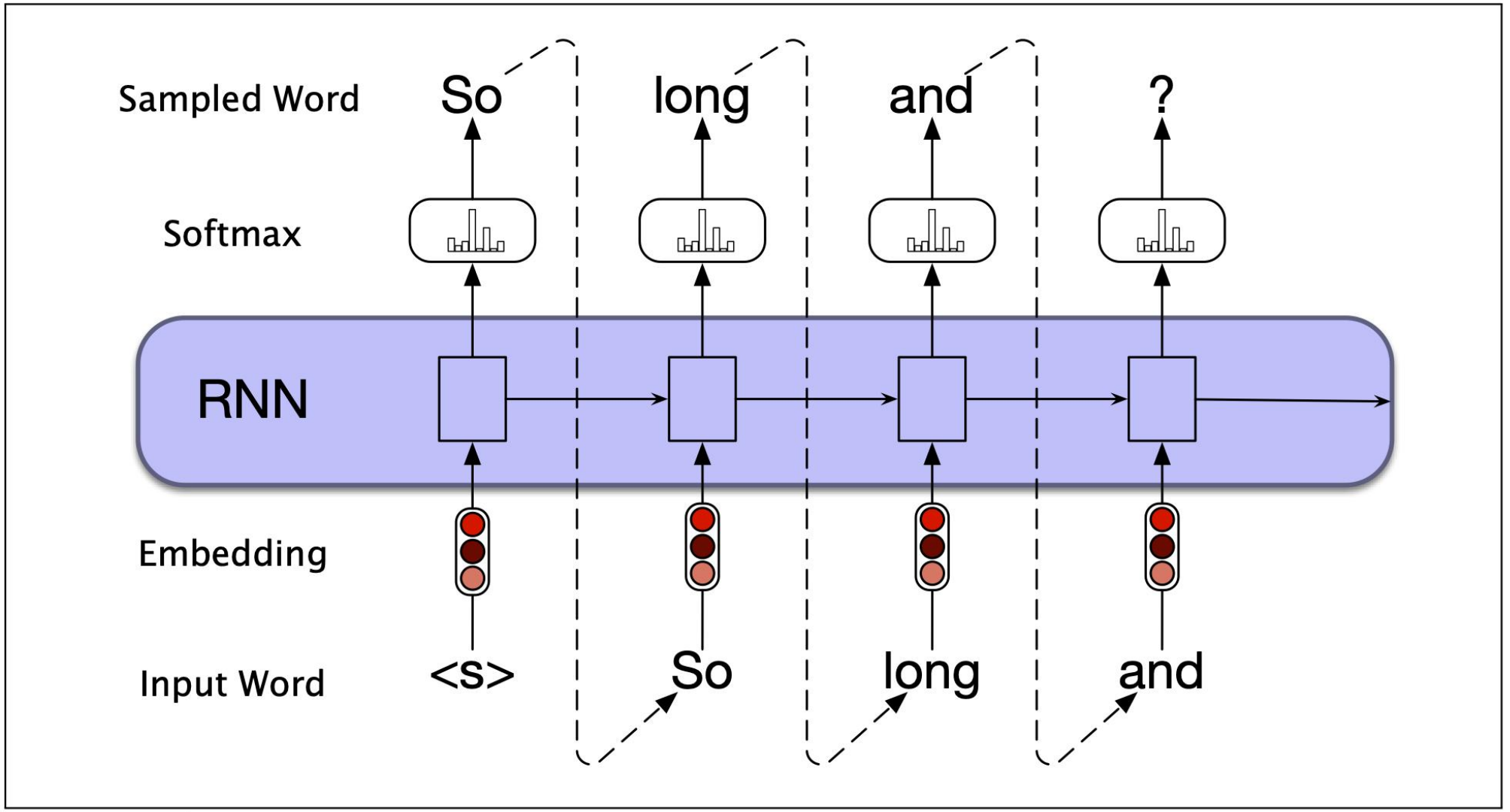


Figure 9.9 Autoregressive generation with an RNN-based neural language model.

Yığın RNN Ağları

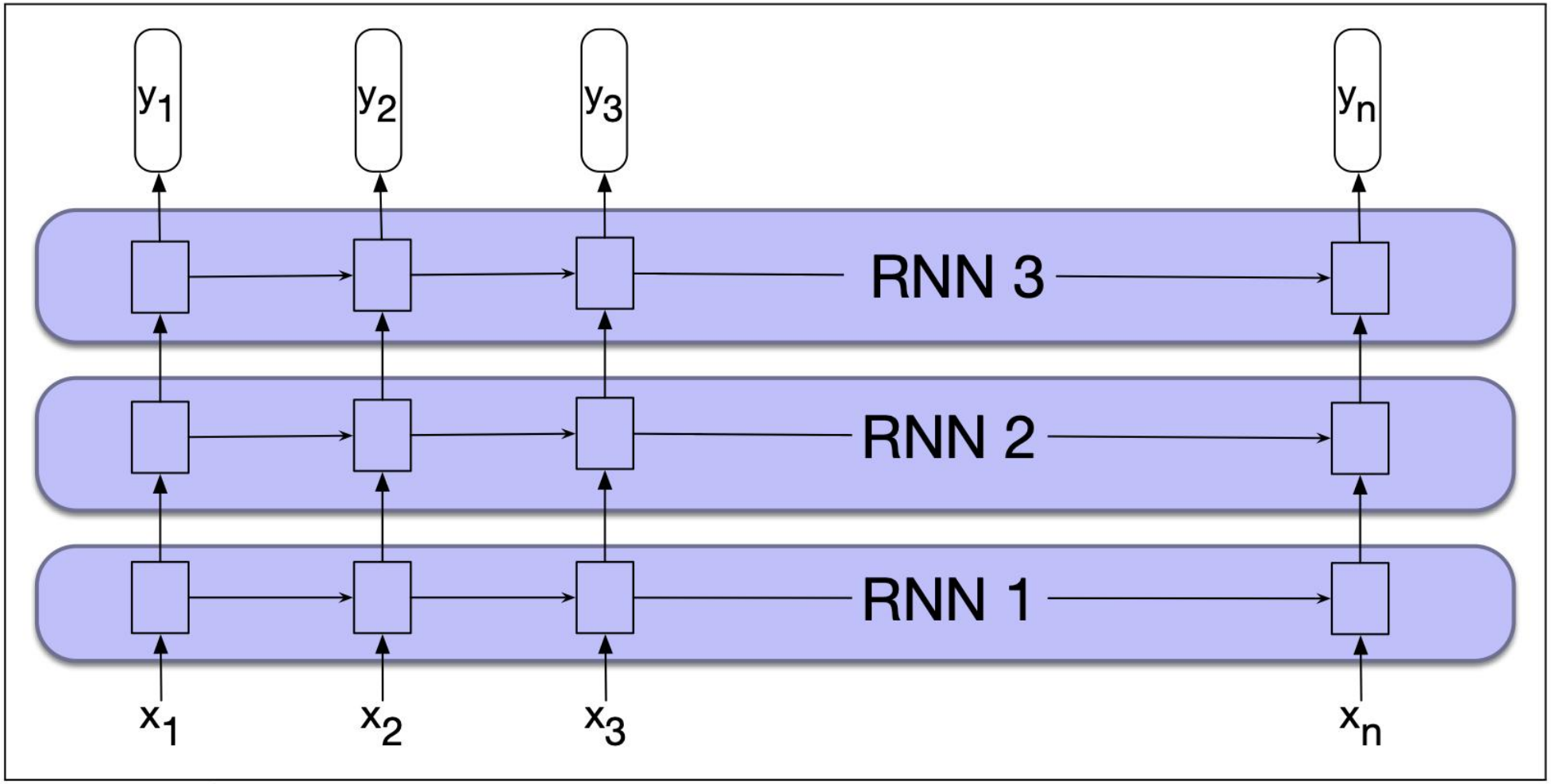


Figure 9.10 Stacked recurrent networks. The output of a lower level serves as the input to higher levels with the output of the last network serving as the final output.

Çift Yönlü RNN Ağları

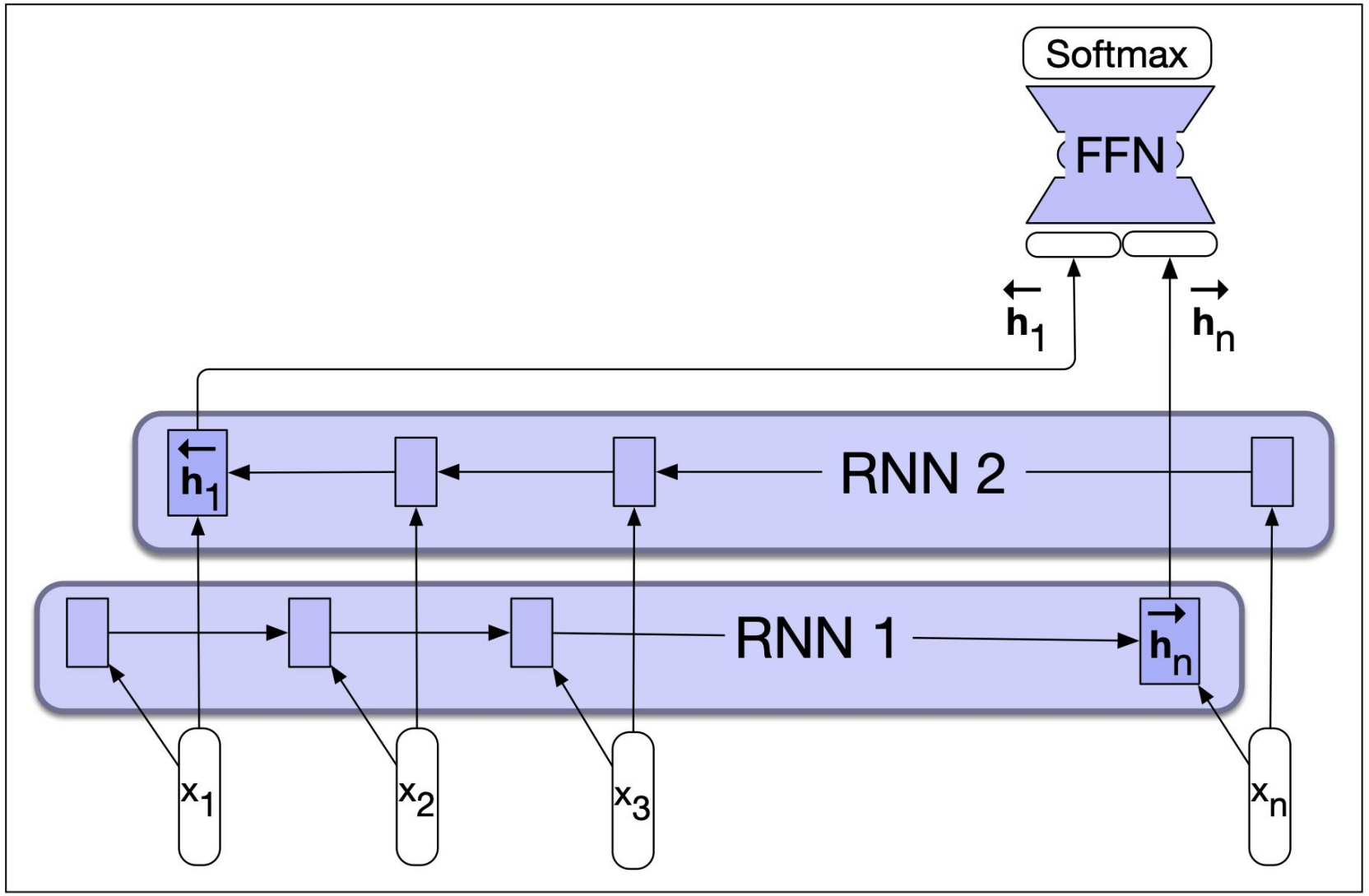


Figure 9.12 A bidirectional RNN for sequence classification. The final hidden units from the forward and backward passes are combined to represent the entire sequence. This combined representation serves as input to the subsequent classifier.

LSTM

- Pratikte, t anındaki yaptığımız işleme uzakta kalan bilgilerin sayısı çok fazlaysa, bunları kullanmayı gerektiren görevler için RNN'leri eğitmek oldukça zordur.
- Önceki dizinin tamamına erişime sahip olmasına rağmen, gizli durumlarda kodlanan bilgiler oldukça yerel olma eğilimindedir. Dolayısıyla girdi dizisinin en son bölümleri ve son kararlarla daha alakalıdır.
- Yine de uzak bilgi, birçok dil uygulaması için kritik öneme sahiptir.
- RNN'lerin kritik bilgileri ileri taşıyamamasının bir nedeni, gizli katmanlardan ve buna bağlı olarak gizli katmandaki değerleri belirleyen ağırlıklardan iki görevi aynı anda gerçekleştirmelerinin istenmesidir:
    - mevcut karar için yararlı bilgiler sağlamak,
    - gelecekteki kararlar için gerekli bilgileri güncellemek ve ileriye taşımak.

- RNN'lerin eğitimiyle ilgili ikinci bir zorluk, hata sinyalini zamanda geriye doğru geri yayma ihtiyacından kaynaklanır (backpropagation through time).
- Bu problemi çözmek için RNN'lerin altındaki en sık kullanılan yapı, uzun kısa süreli bellek (LSTM) ağıdır (Hochreiter and Schmidhuber, 1997).

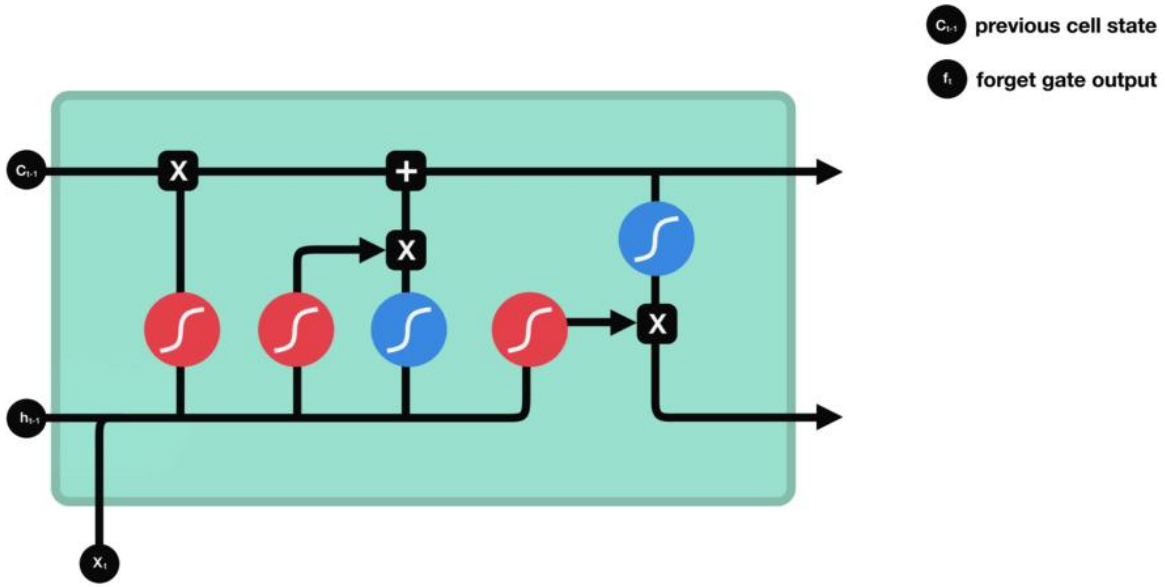
- LSTM'ler, içerik yönetimi problemini iki alt probleme ayırır:
  1) artık ihtiyaç duyulmayan bilgileri bağlamdan çıkarmak ve
  2) daha sonraki karar verme için ihtiyaç duyulabilecek bilgileri eklemek.

  Her iki sorunu da çözmenin anahtarı, mimariye bir strateji kodlamak yerine bu bağlamın nasıl yönetileceğini öğrenmektir.

  LSTM'ler bunu, önce mimariye açık bir bağlam katmanı ekleyerek (olağan yinelenen gizli katmana ek olarak) ve birimlere giren ve çıkan bilgi akışını kontrol etmek için kapılardan yararlanan özel sinir birimlerinin kullanımıyla başarır. Bu kapılar, giriş ve önceki gizli katman ve önceki bağlam katmanları üzerinde sırayla çalışan ek ağırlıkların kullanılmasıyla gerçekleştirilir.

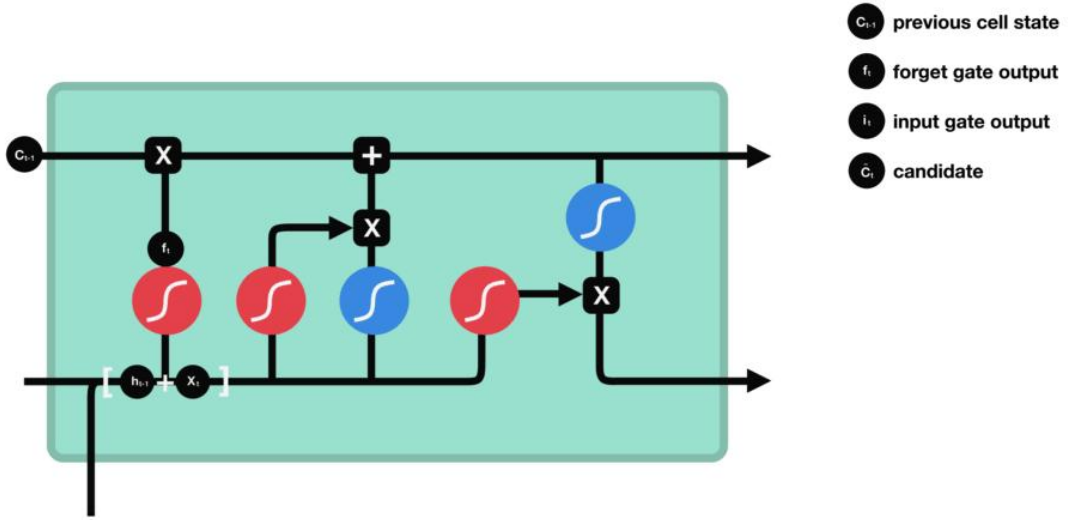
Forget Gate (Unutma Kapısı)

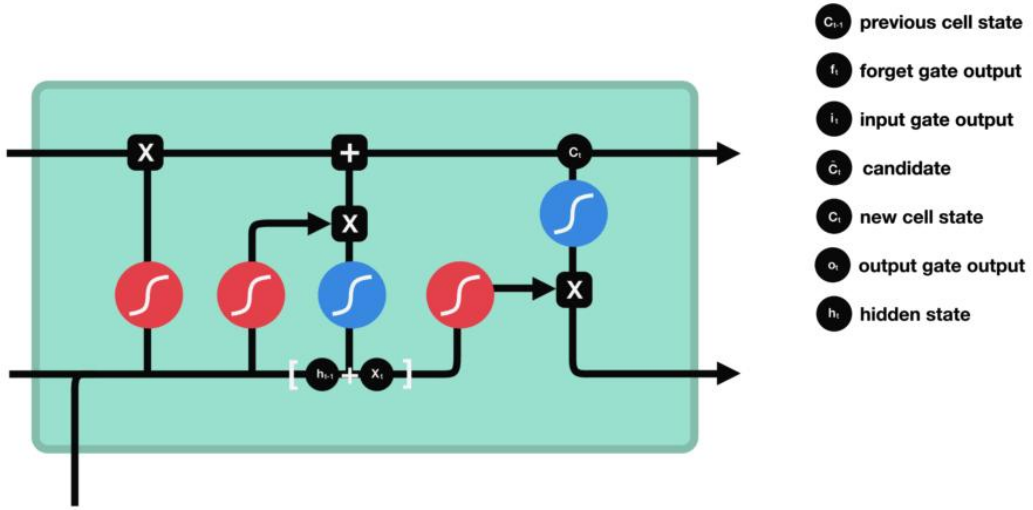
- Bu geçidin amacı, artık ihtiyaç duyulmayan bağlamdan bilgi silmektir.
- Unutma kapısı, önceki durumun gizli katmanının ve mevcut girdinin ağırlıklı bir toplamını hesaplar ve bunu bir sigmoidden geçirir. Bu maske daha sonra, artık gerekli olmayan bağlamdan bilgileri çıkarmak için içerik vektörü ile öğe bazında çarpılır.
- Önceki dizinin tamamına erişime sahip olmasına rağmen, gizli durumlarda kodlanan bilgiler oldukça yerel olma eğilimindedir. Dolayısıyla girdi dizisinin en son bölümleri ve son kararlarla daha alakalıdır.

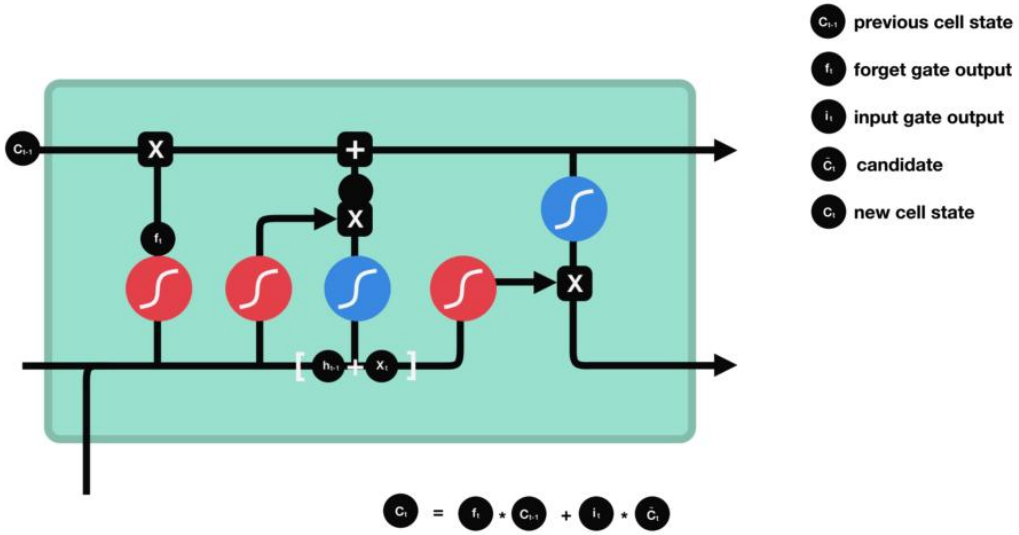


Input Gate (Girdi Kapısı)

- Cell State'i güncellemek için kullanılır. Sigmoid fonksiyonu uygulanır, hangi bilginin tutulacağına karar verilir.
- Ağı düzenlemek için de Tanh fonksiyonu yardımıyla -1,1 arasına indirgenir ve çıkan iki sonuç çarpılır.







Output Gate (Çıktı Kapısı)

- Bir sonraki katmana gönderilecek değere karar verir. Bu değer, tahmin için kullanılır.
- Bir önceki değer ile şu anki girdi Sigmoid fonksiyonundan geçer.
- Cell State'den gelen değer, Tanh fonksiyonundan geçtikten sonra iki değer çarpılır ve bir sonraki katmana "Bir önceki değer" olarak gider. Cell State ilerler.

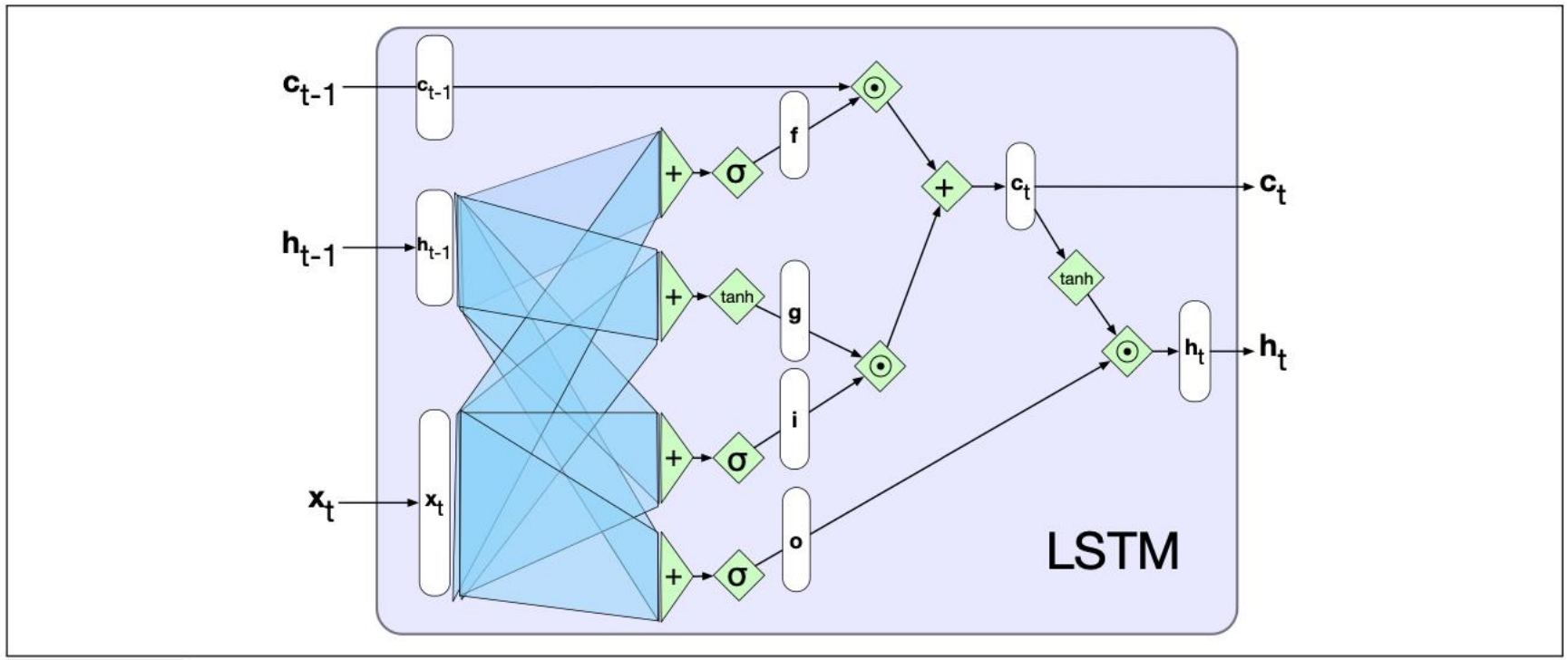


Figure 9.13 A single LSTM unit displayed as a computation graph. The inputs to each unit consists of the current input, x, the previous hidden state, h_{t-1}, and the previous context, c_{t-1}. The outputs are a new hidden state, h_t and an updated context, c_t.

Öz-Dikkat Ağları (Self-Attention): Transformers Yaklaşımı

- Kapıların eklenmesi, LSTM'lerin RNN'lere oranla daha uzaktaki bilgileri işlemesine izin verse de, altta yatan sorunu tam olarak çözmezler: bilginin uzun bir dizi olarak yinelenen bağlantıdan geçirilmesi, bilgi kaybına ve eğitimde zorluklara yol açar.
- Ayrıca, tekrarlayan ağların doğası gereği sıralı yapısı paralel olarak hesaplama yapmayı zorlaştırır.

- Transformers, basit doğrusal katmanları, ileri beslemeli ağları ve bu yapının temel yeniliği olan öz-dikkat (self-attention) katmanlarını birleştirerek yapılan çok katmanlı ağlar olan transformer blok yığınlarından oluşur.

- Öz dikkat (self-attention), bir ağın RNN'lerde olduğu gibi aralardaki tekrarlayan bağlantılardan geçmesine gerek kalmadan, keyfi olarak büyük bağlamlardan bilgileri doğrudan çıkarmasına ve kullanmasına izin verir.

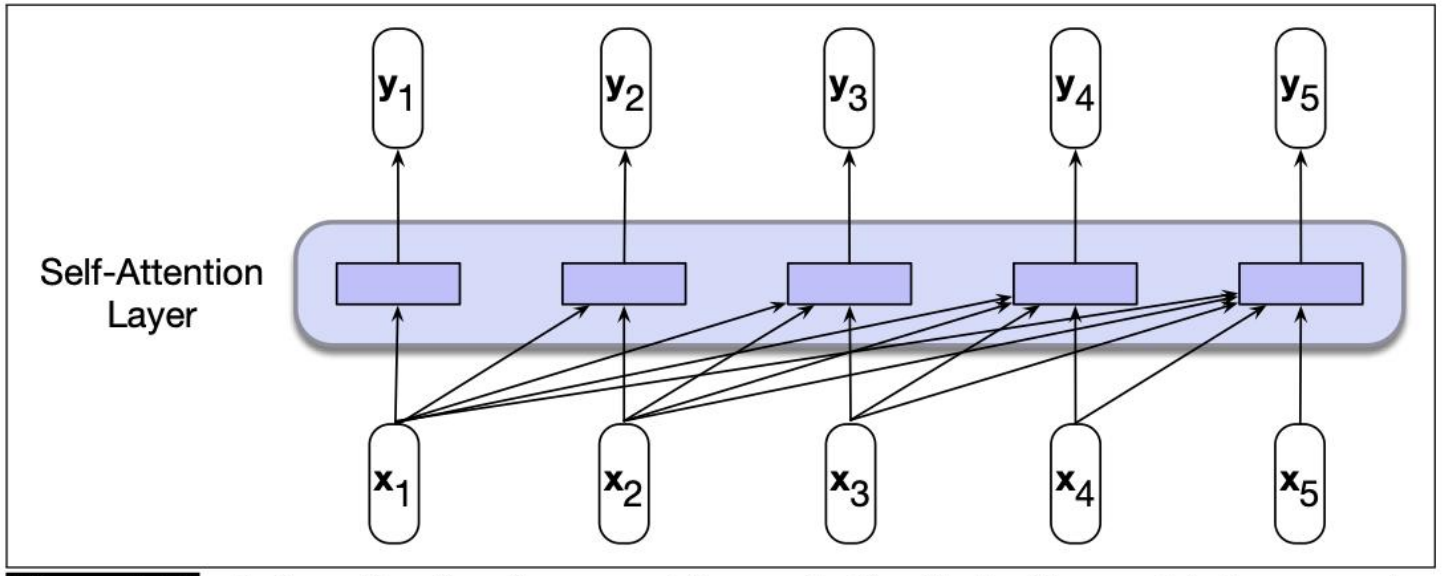


Figure 9.15 Information flow in a causal (or masked) self-attention model. In processing each element of the sequence, the model attends to all the inputs up to, and including, the current one. Unlike RNNs, the computations at each time step are independent of all the other steps and therefore can be performed in parallel.

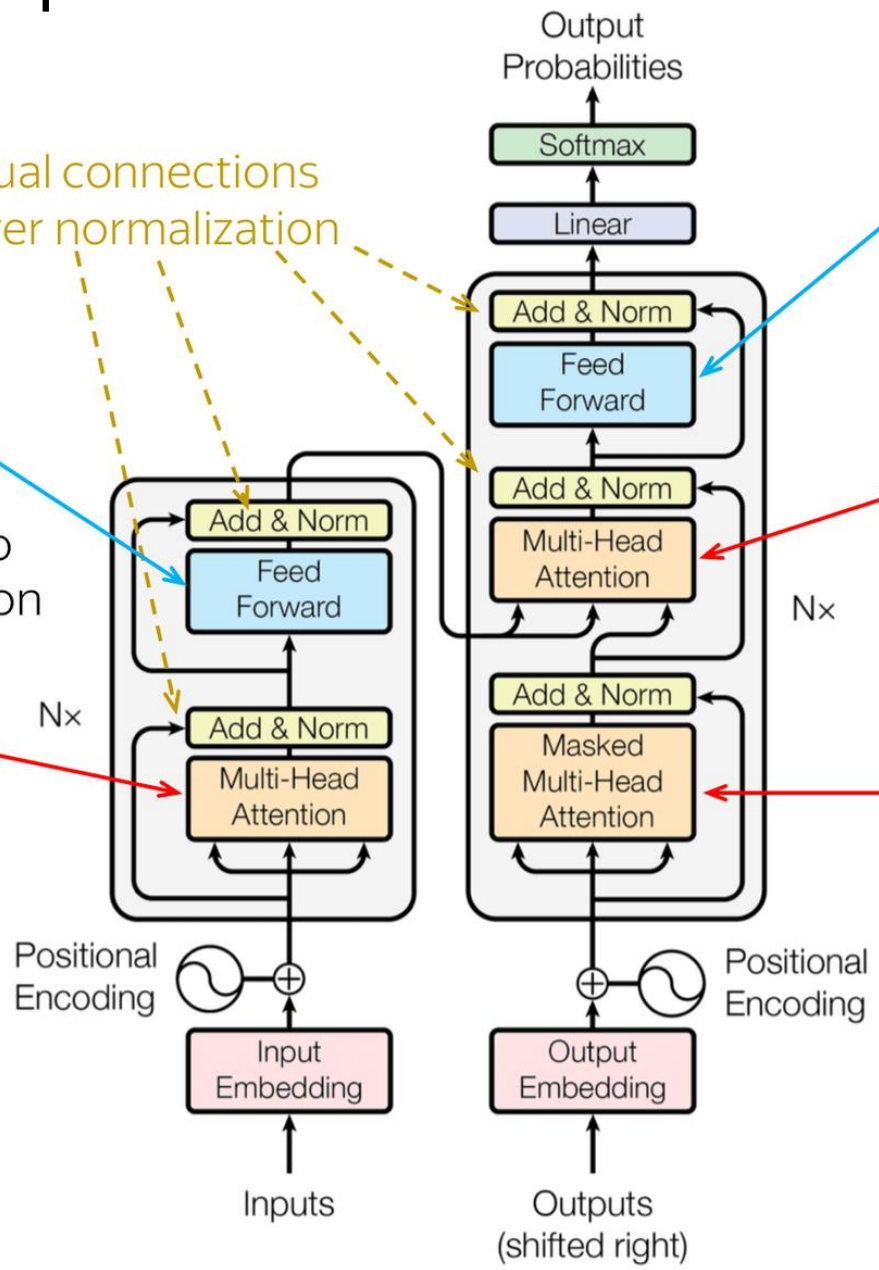
Transformers Yapısı

Residual connections and layer normalization

Feed-forward network: after taking information from other tokens, take a moment to think and process this information

Encoder self-attention: tokens look at each other

queries, keys, values are computed from encoder states



Feed-forward network: after taking information from other tokens, take a moment to think and process this information

Decoder-encoder attention: target token looks at the source

queries – from decoder states; keys and values from encoder states

Decoder self-attention (masked): tokens look at the previous tokens

queries, keys, values are computed from decoder states

Self-Attention Yaklaşımı

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

$A(q, K, V)$= attention-based vector representation of a word

calculate for each word

Softmax Fonksiyonu

input vektörü (z0, ... zK)

negatif inputları pozitif aralığa çekmek için standart exp fonksiyonu

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

class sayısı

normalizasyon işlemi

RNN Attention

$$\alpha^{<t,t'>} = \frac{\exp(e^{<t,t'>})}{\sum_{t'=1}^{T_x} \exp(e^{<t,t'>})}$$

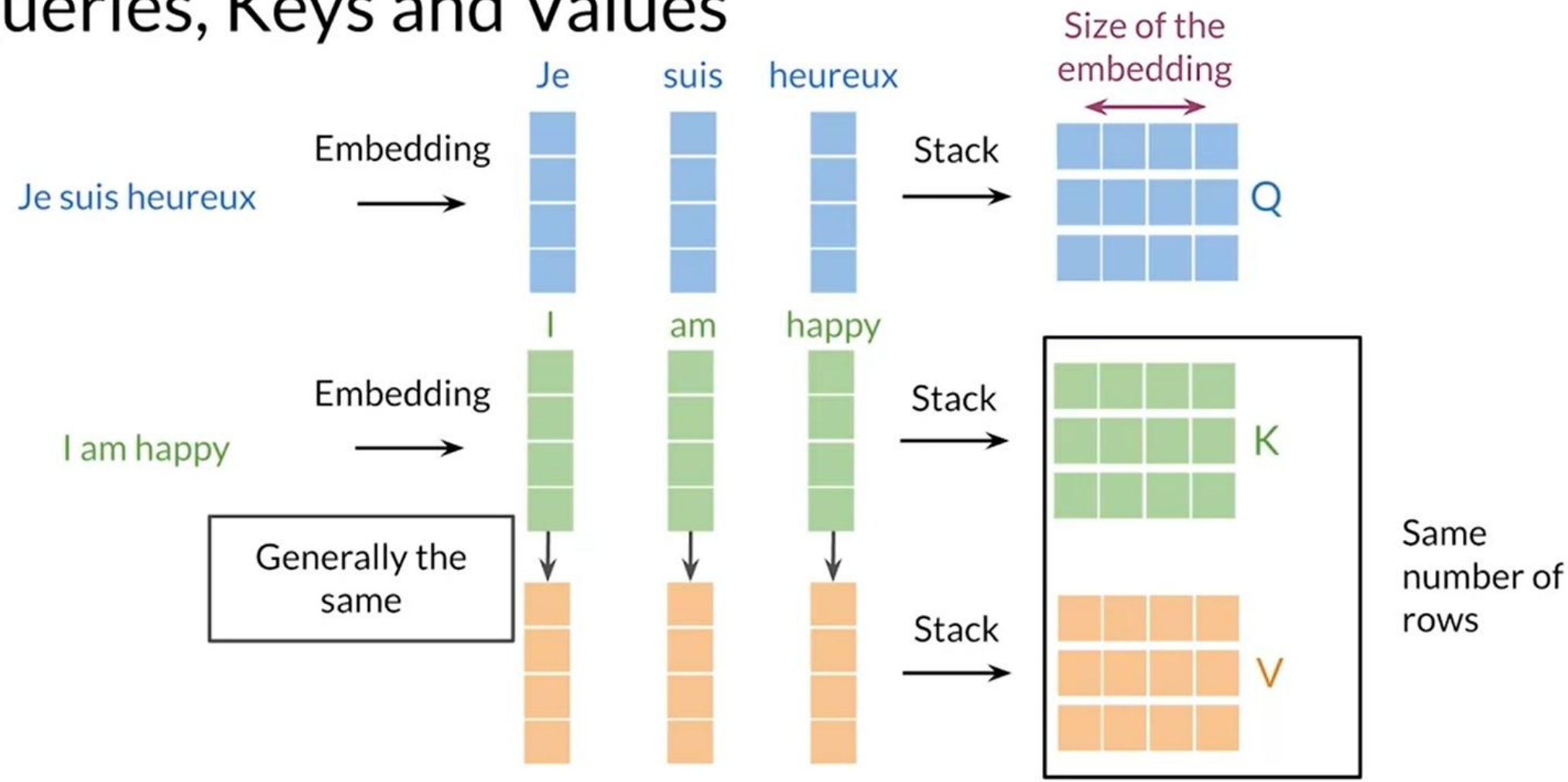
Transformers Attention

$$A(q, K, V) = \sum_i \frac{\exp(e^{<q \cdot k^{<i>}>})}{\sum_j \exp(e^{<q \cdot k^{<j>}>})} v^{<i>}$$

Q = bir cümledeki kelimelerle ilgili ilginç sorular,
K = Q verilen kelimelerin nitelikleri,
V = Q verilen kelimelerin belirli temsilleri

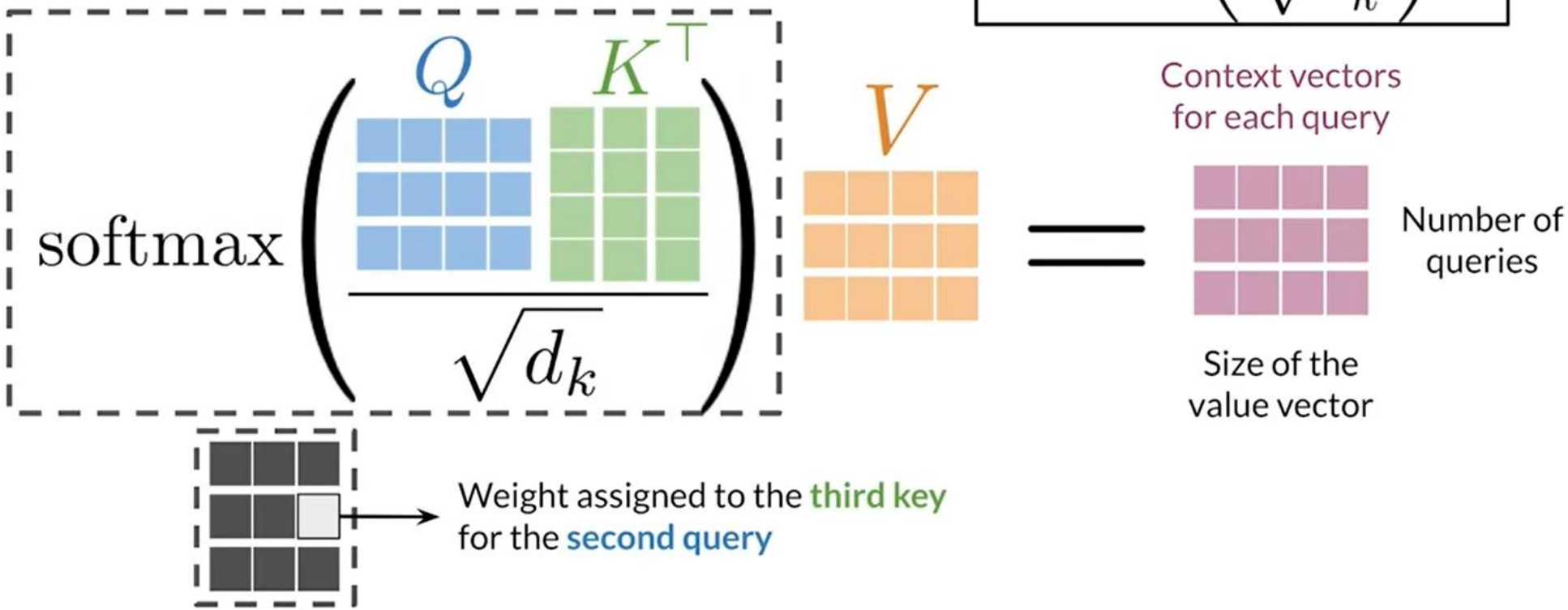
$x^{<1>}$	$x^{<2>}$	$x^{<3>}$	$x^{<4>}$	$x^{<5>}$
Jane	visite	l'Afrique	en	septembre

Queries, Keys and Values



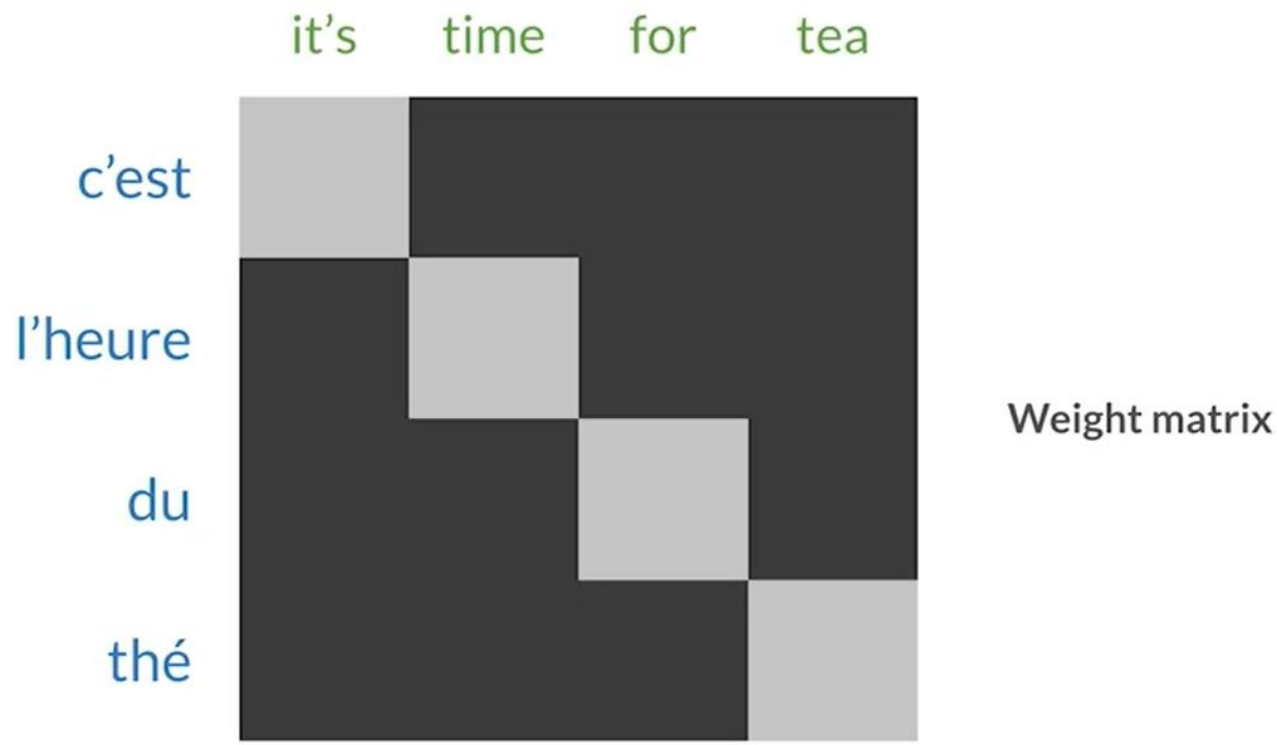
Attention Math

$$\text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$



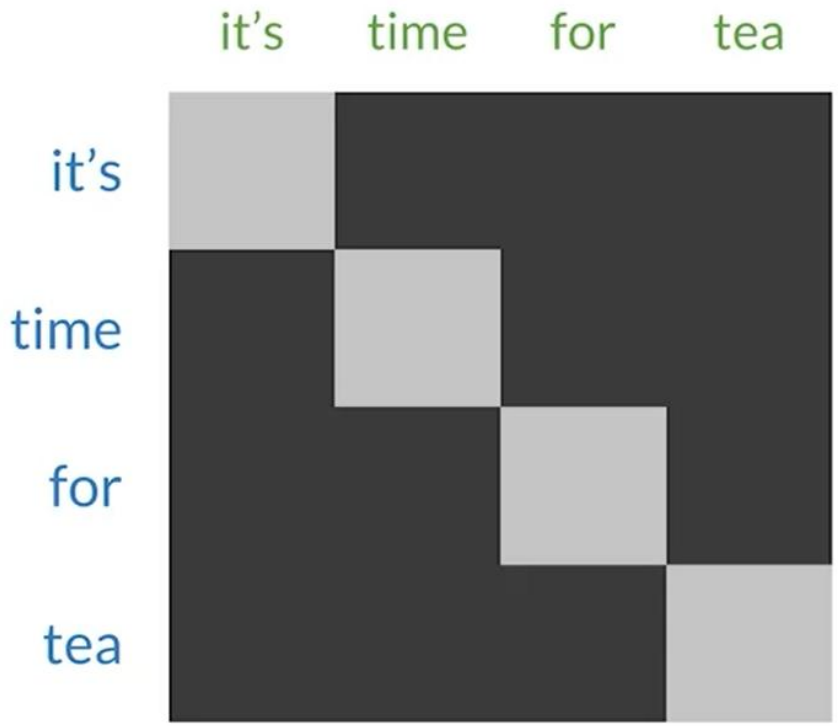
Encoder-Decoder Attention

Queries from one sentence, keys and values from another



Self-Attention

Queries, keys and values come from the **same sentence**

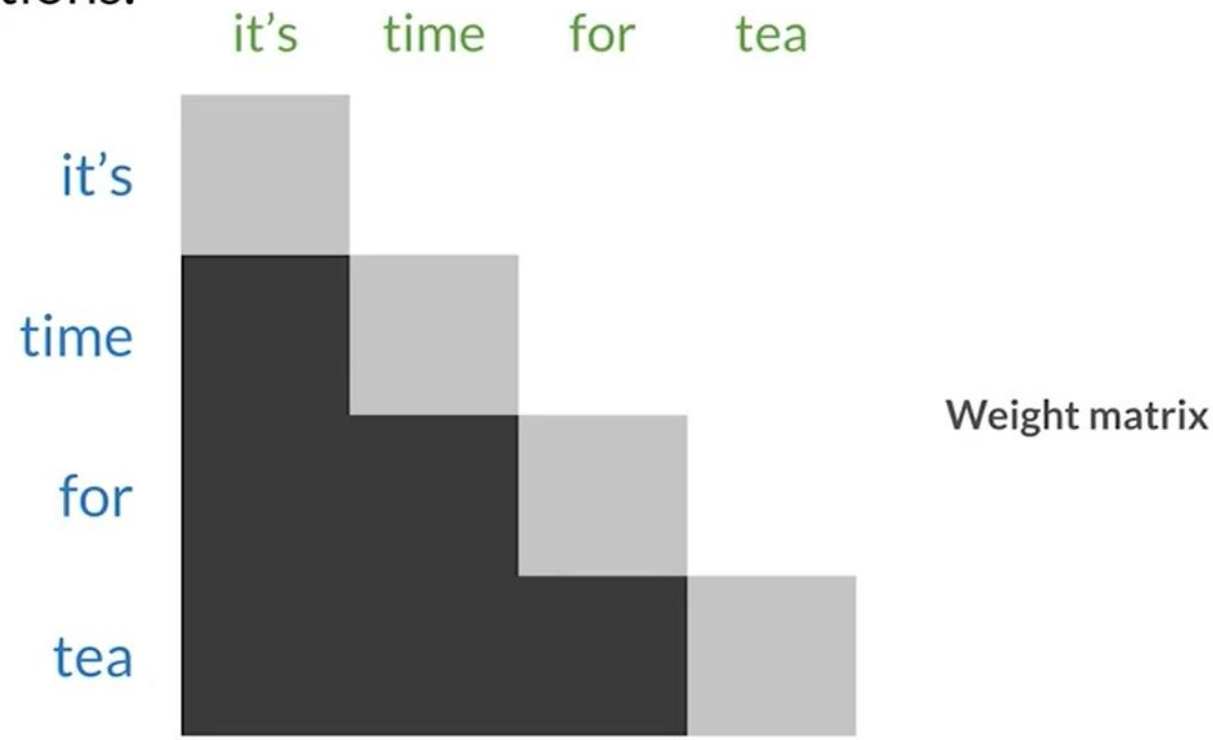


Weight matrix

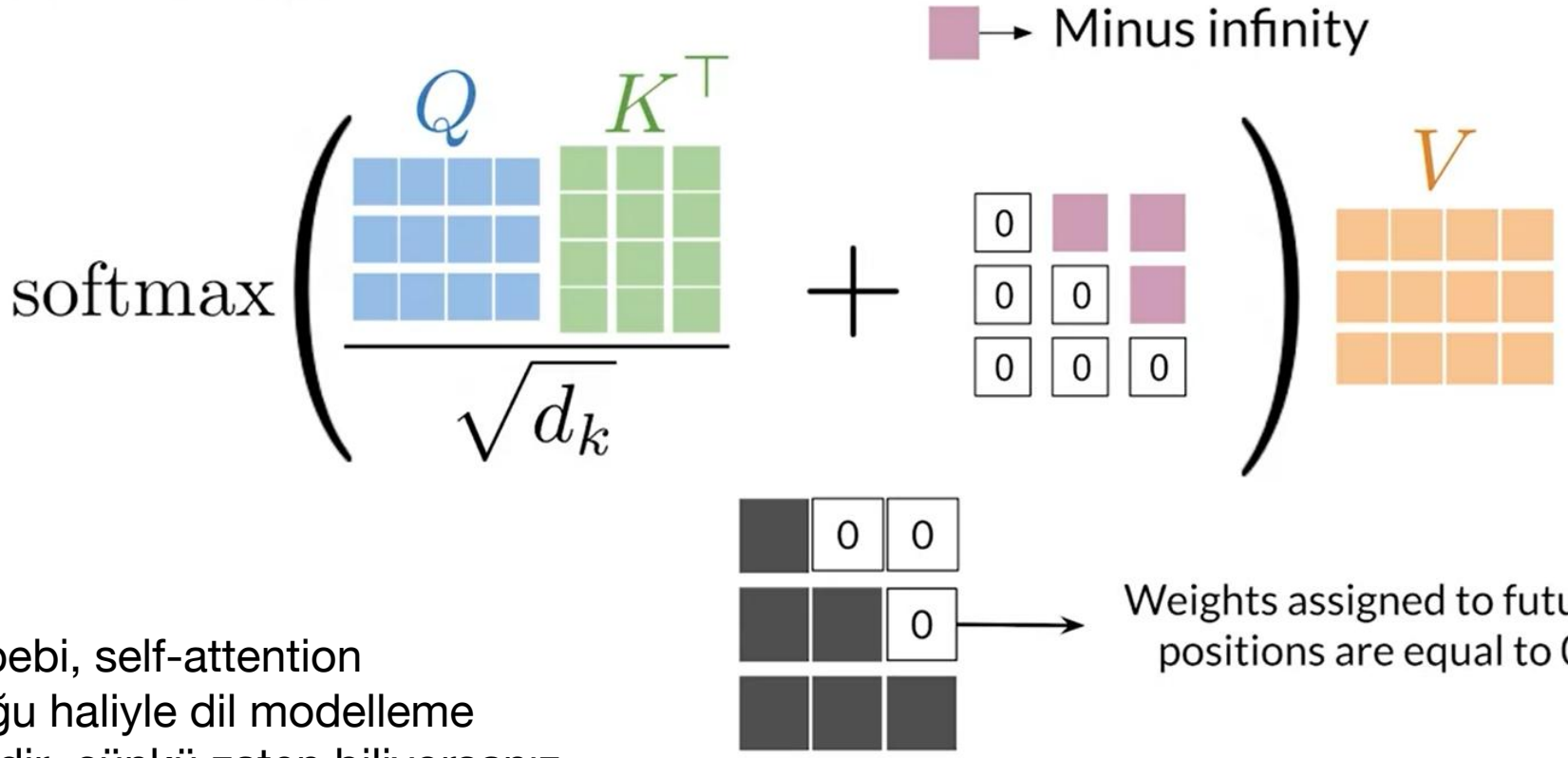
Meaning of each word **within** the sentence

Masked self-attention math

Queries, keys and values come from the **same sentence**. Queries don't attend to future positions.



Masked self-attention math



- Bunun yapılmasının sebebi, self-attention formülündeki QK^T olduğu haliyle dil modelleme ortamında anlamlı değildir, çünkü zaten biliyorsanız bir sonraki kelimeyi tahmin etmek oldukça basittir. Bunu düzeltmek için, matrisin üst üçgen kısmındaki öğeler sıfırlanır ($-\infty$'a ayarlanır), böylece dizide takip eden kelimelerin herhangi bir bilgisi varsa ortadan kaldırılır.

Masked self-attention math

- Yandaki figür, dikkatin girdi uzunluğunda ikinci dereceden olduğunu açıkça ortaya koymaktadır, çünkü her katmanda, girdideki her token çifti arasındaki nokta çarpımları (dot product) hesaplamamız gerekir. Bu, bir transformer'a girişin uzun belgelerden (tüm Wikipedia sayfaları veya romanlar gibi) oluşmasını son derece maliyetli hale getirir ve bu nedenle çoğu uygulamanın giriş uzunluğunu, örneğin en fazla bir sayfa veya bir metin paragrafı ile sınırlaması gerekir.
- Daha verimli dikkat mekanizmaları bulmak, devam eden bir araştırma alanıdır.

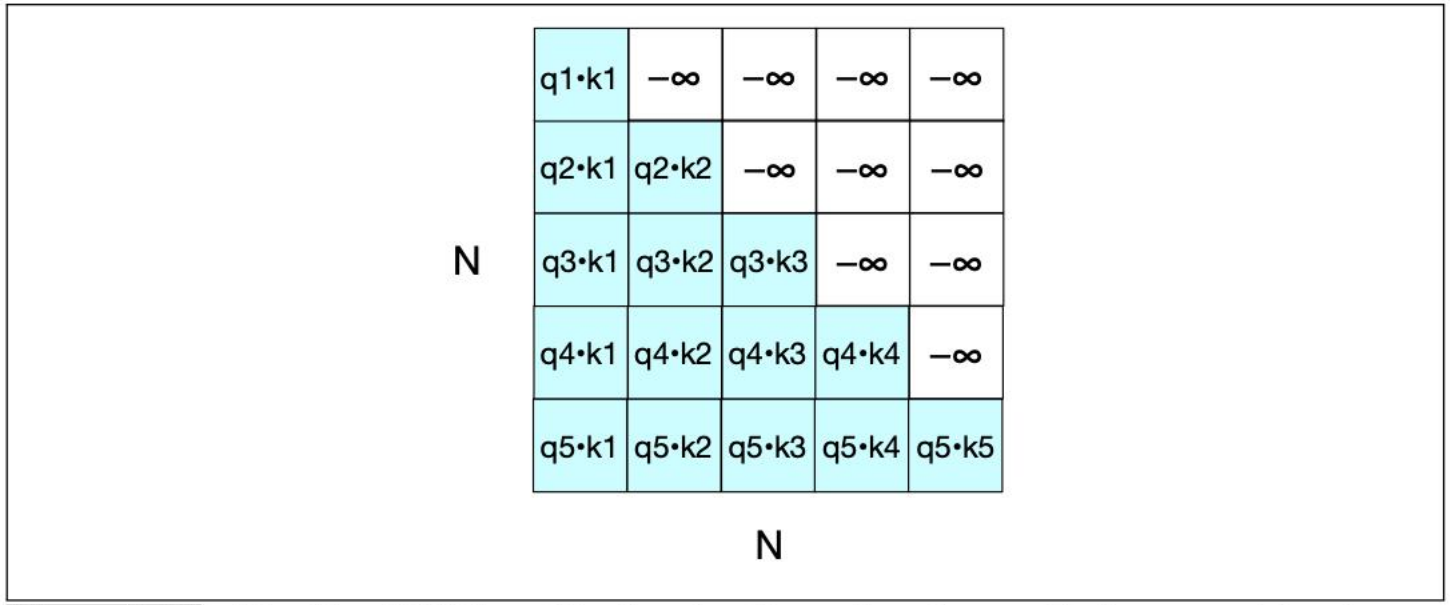
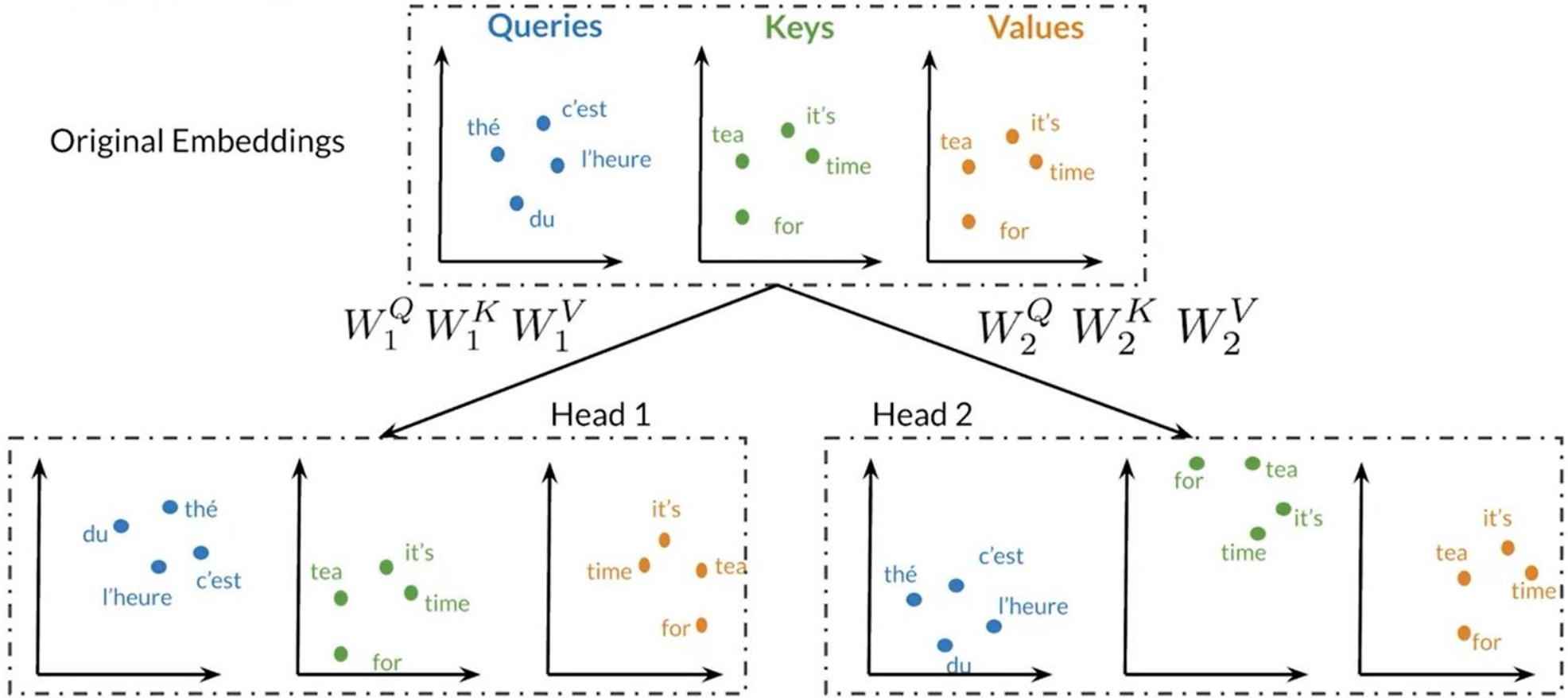
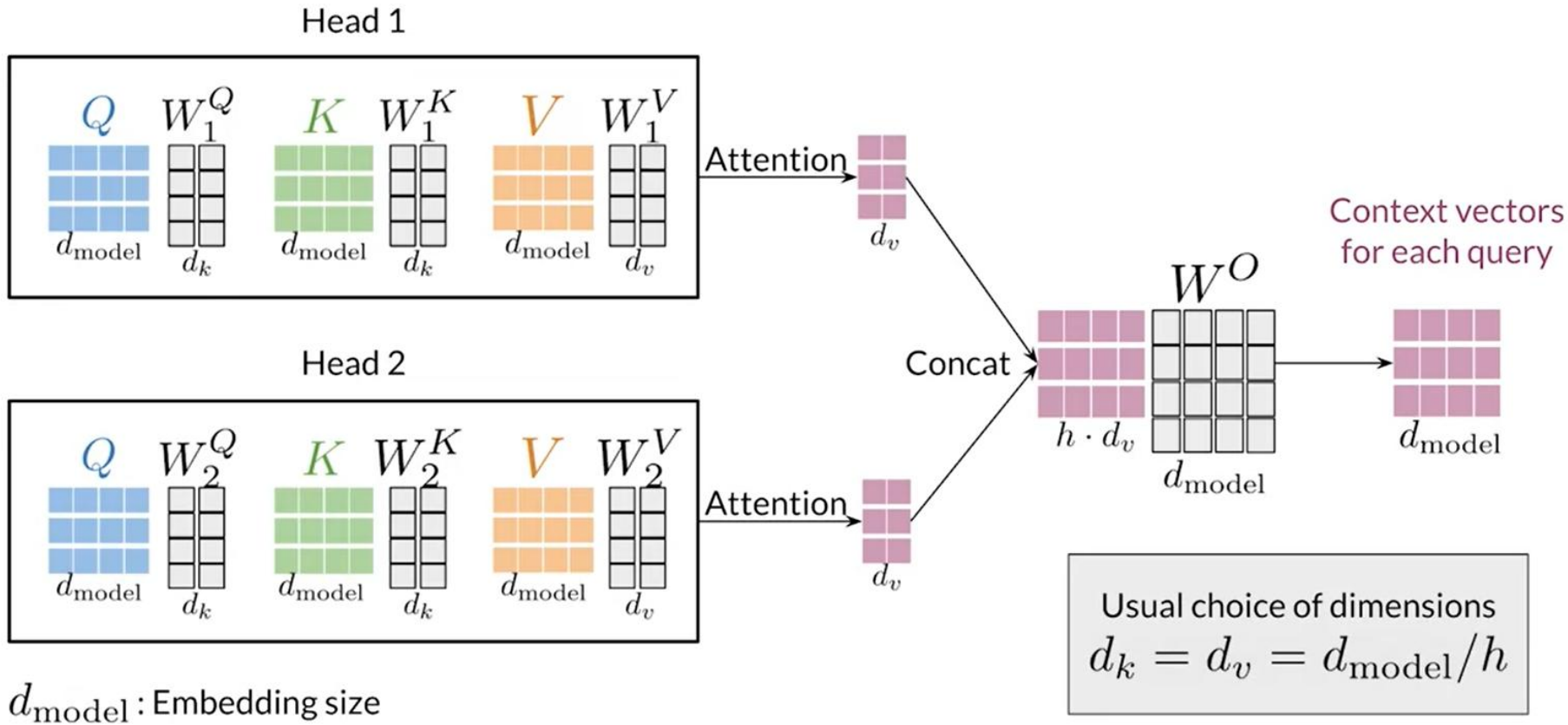


Figure 9.17 The $N \times N$ $\mathbf{QT}^\intercal$ matrix showing the $q_i \cdot k_j$ values, with the upper-triangle portion of the comparisons matrix zeroed out (set to $-\infty$, which the softmax will turn to zero).

Multi-Head Attention - Overview



Multi-Head Attention



d_{model} : Embedding size

Usual choice of dimensions

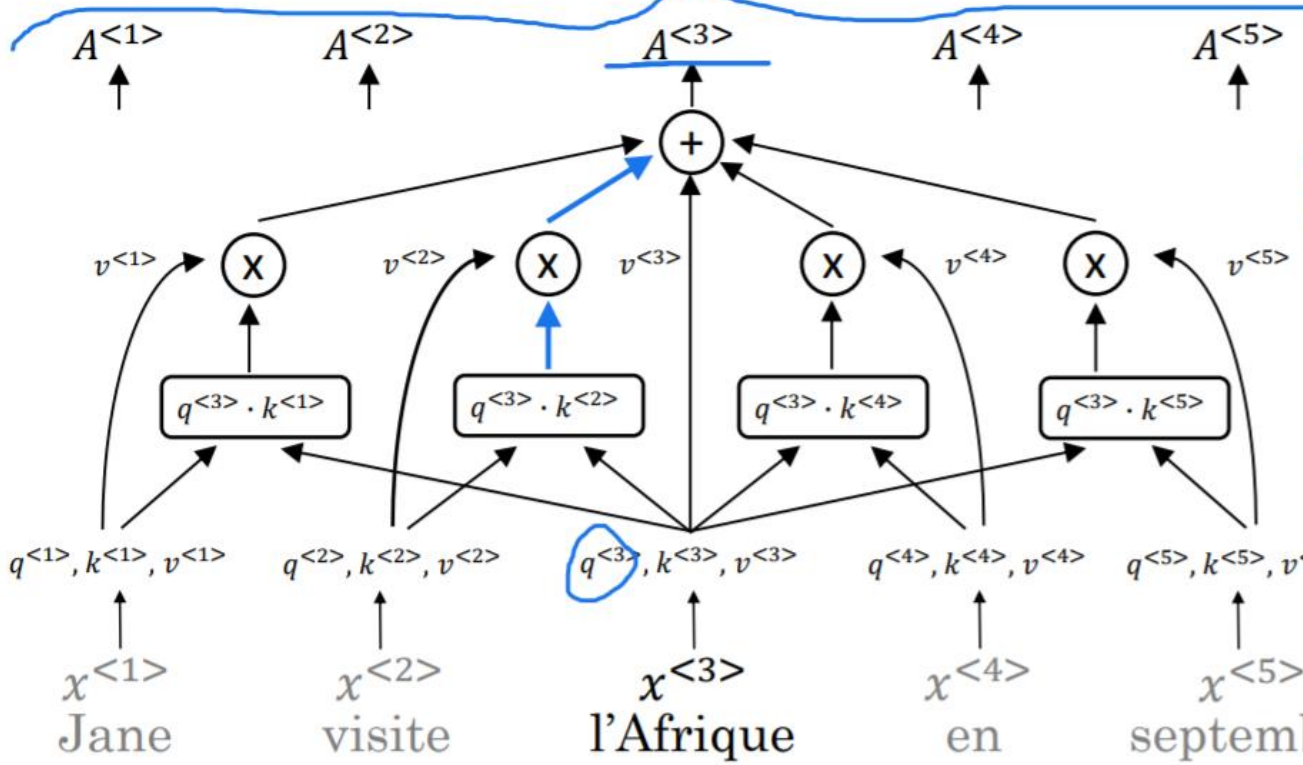
$$d_k = d_v = d_{\text{model}}/h$$

Self-Attention

$$A(q, K, V) = \sum_i \frac{\exp(e^{\langle q \cdot k^{<i>} \rangle})}{\sum_j \exp(e^{\langle q \cdot k^{<j>} \rangle})} v^{<i>}$$

softmax

$$\textbf{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \textbf{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{\boldsymbol{d}_k}}\right)\boldsymbol{V}$$



Did what?

Query (Q)	Key (K)	Value (V)
$q^{<1>}$	$k^{<1>}$	$v^{<1>}$
$q^{<2>}$	$k^{<2>}$	$v^{<2>}$
$q^{<3>}$	$k^{<3>}$	$v^{<3>}$
$q^{<4>}$	$k^{<4>}$	$v^{<4>}$
$q^{<5>}$	$k^{<5>}$	$v^{<5>}$

person action — Jane visit

What's happening there?

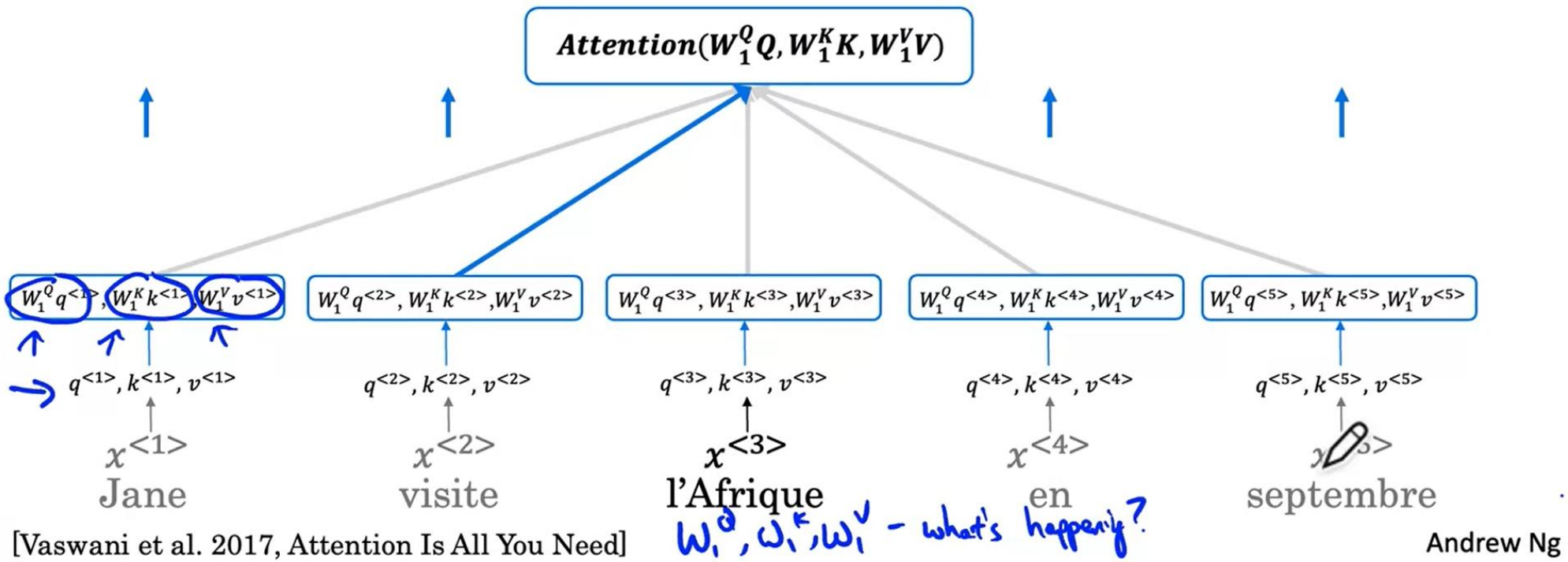
W^Q, W^K, W^V

Multi-Head Attention

"head"

$$MultiHead(Q,K,V)$$

$$\textbf{Attention}(Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

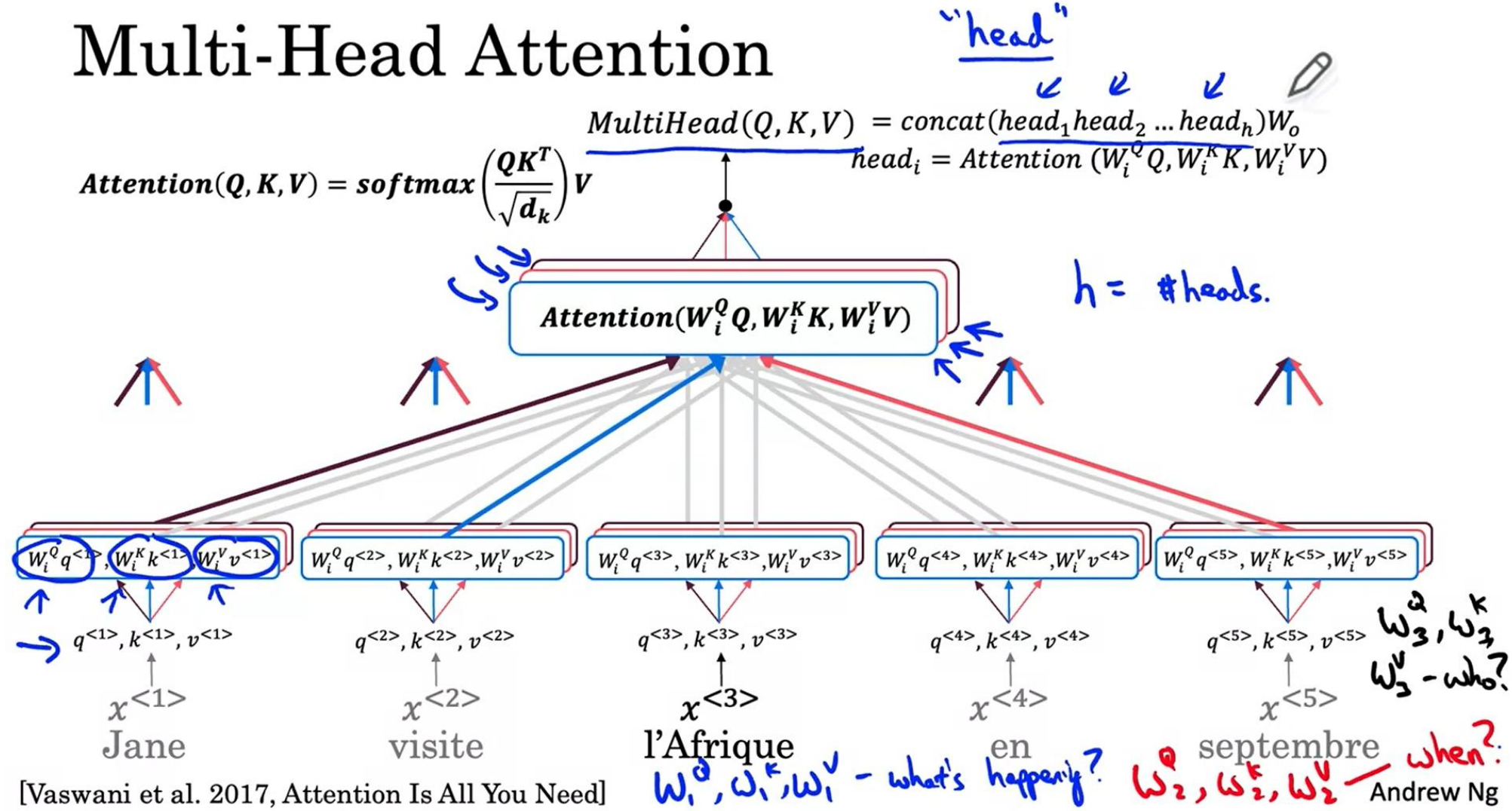


[Vaswani et al. 2017, Attention Is All You Need]

W_1^Q, W_1^K, W_1^V - what's happening?

Andrew Ng

Multi-Head Attention



[Vaswani et al. 2017, Attention Is All You Need]

Andrew Ng

- Daha iyi anlaşılması için Multi-head Attention'ın başka bir gösterimi.

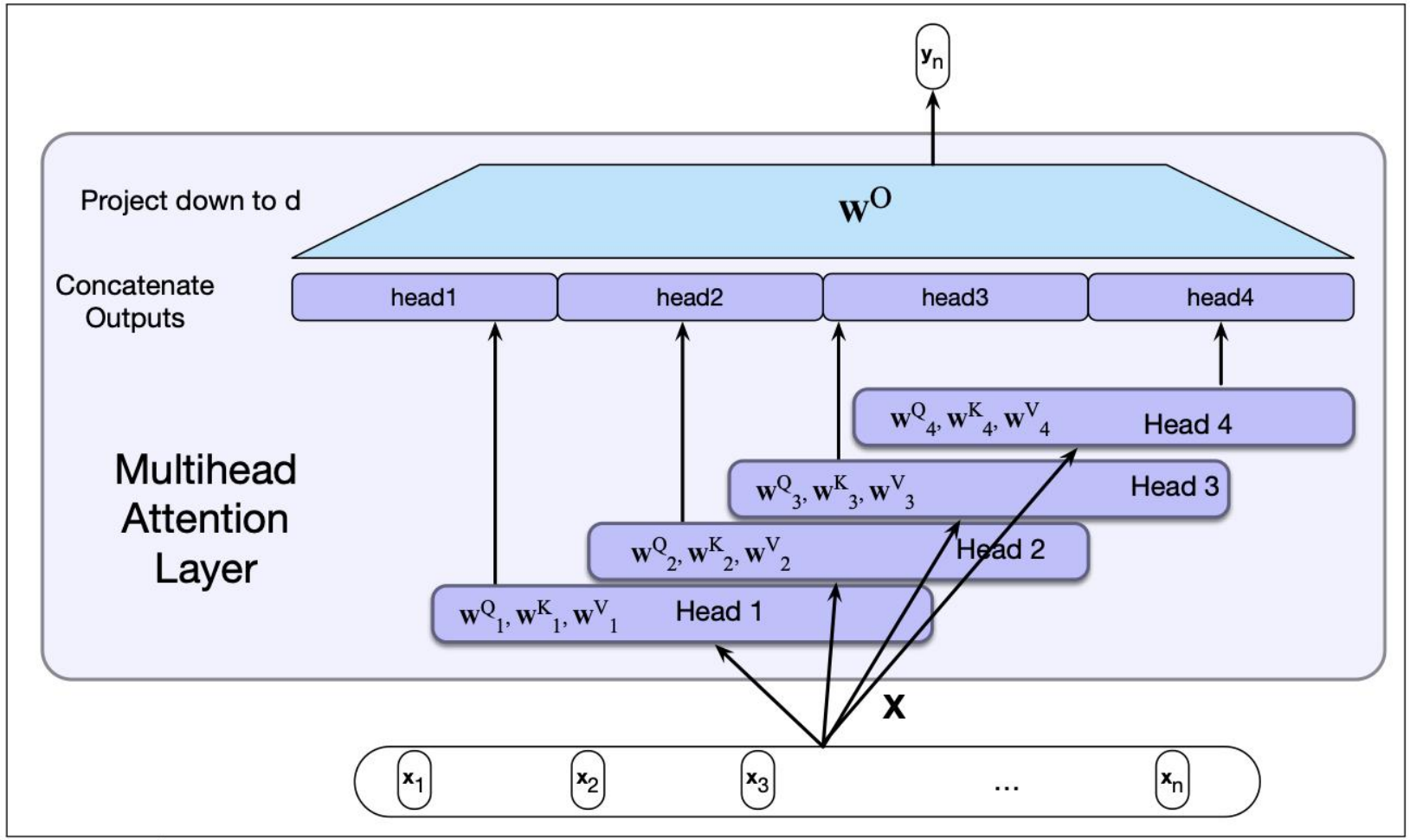
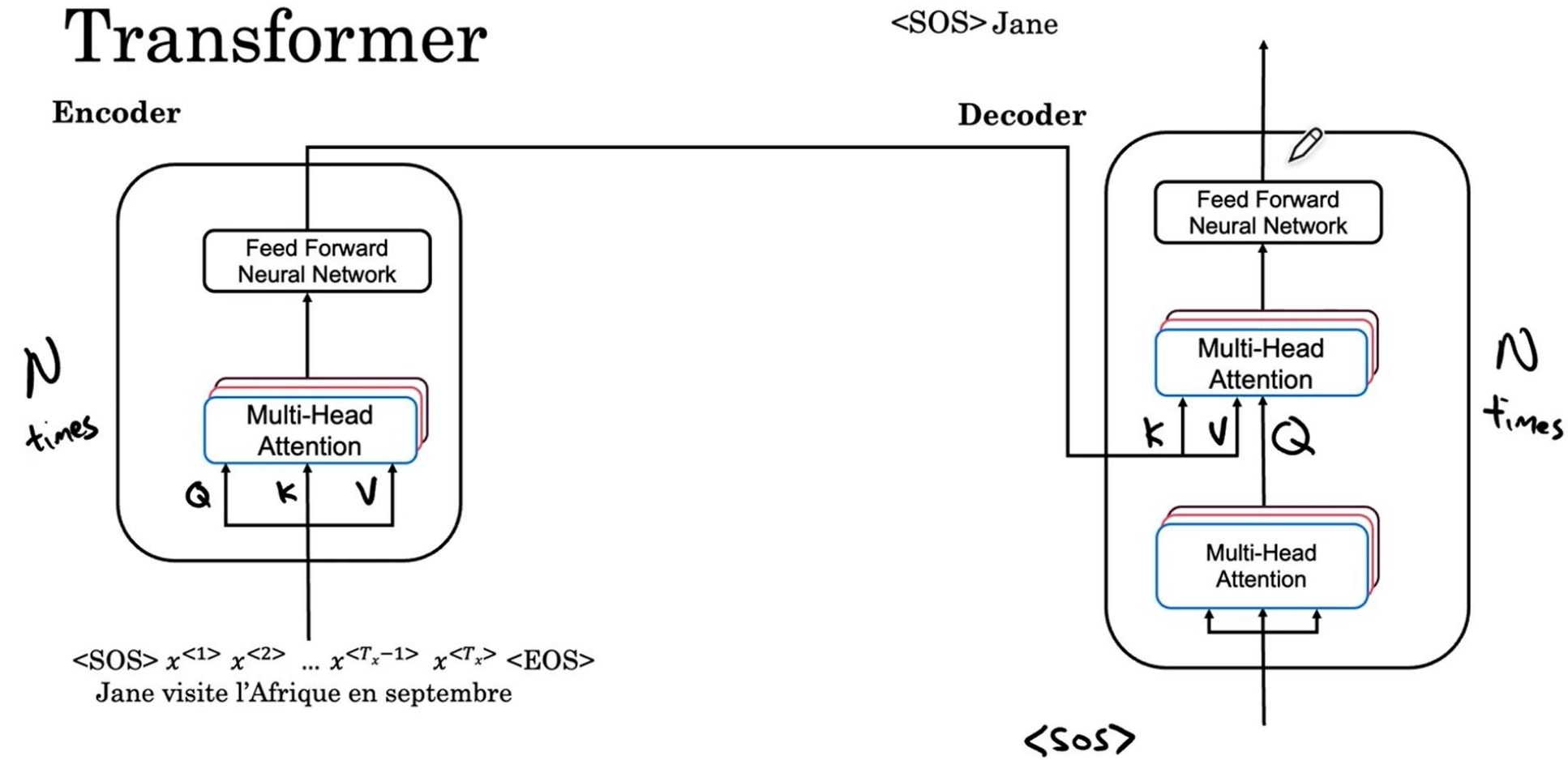


Figure 9.19 Multihead self-attention: Each of the multihead self-attention layers is provided with its own set of key, query and value weight matrices. The outputs from each of the layers are concatenated and then projected down to d, thus producing an output of the same size as the input so layers can be stacked.

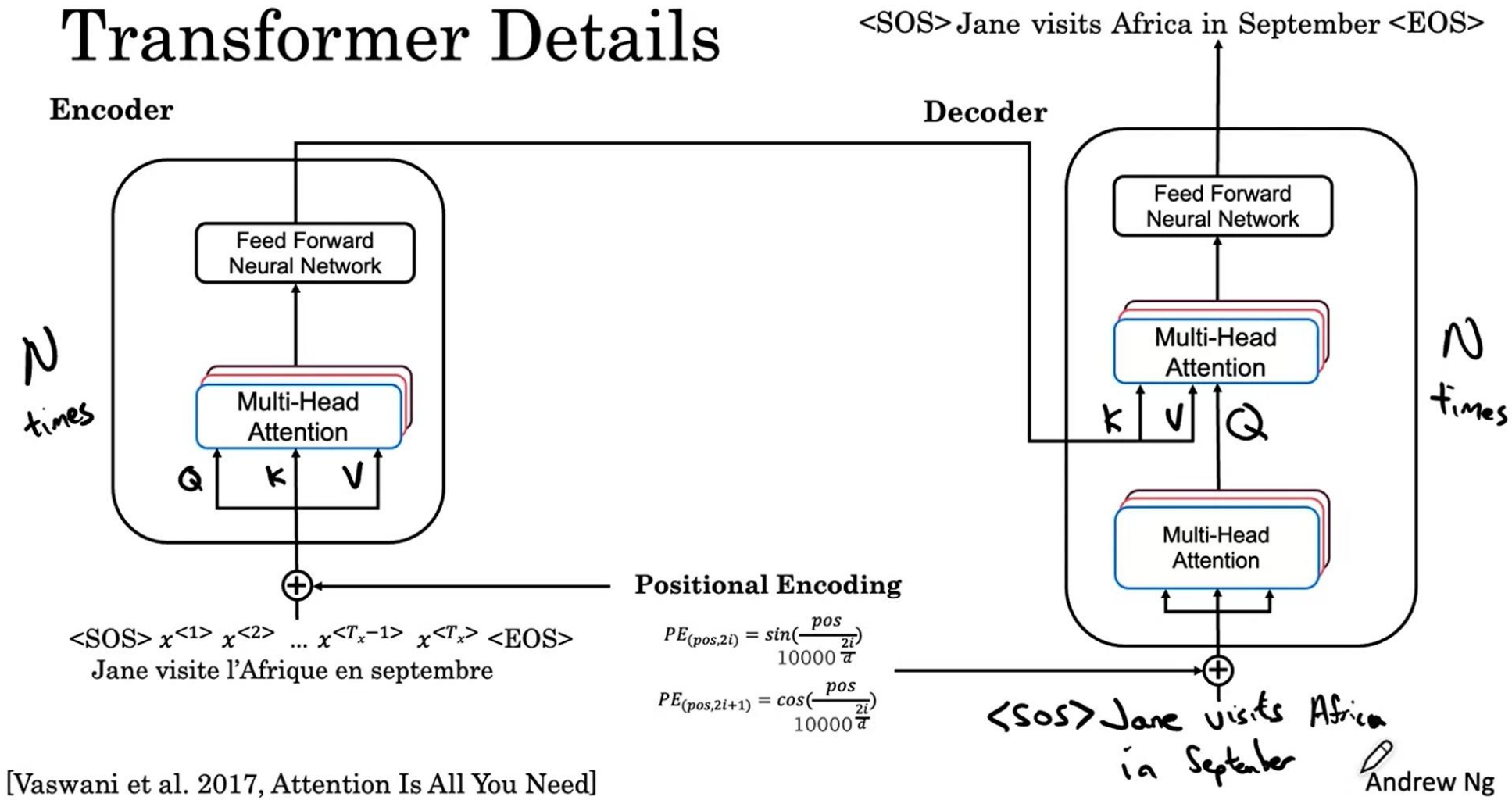
Transformer



[Vaswani et al. 2017, Attention Is All You Need]

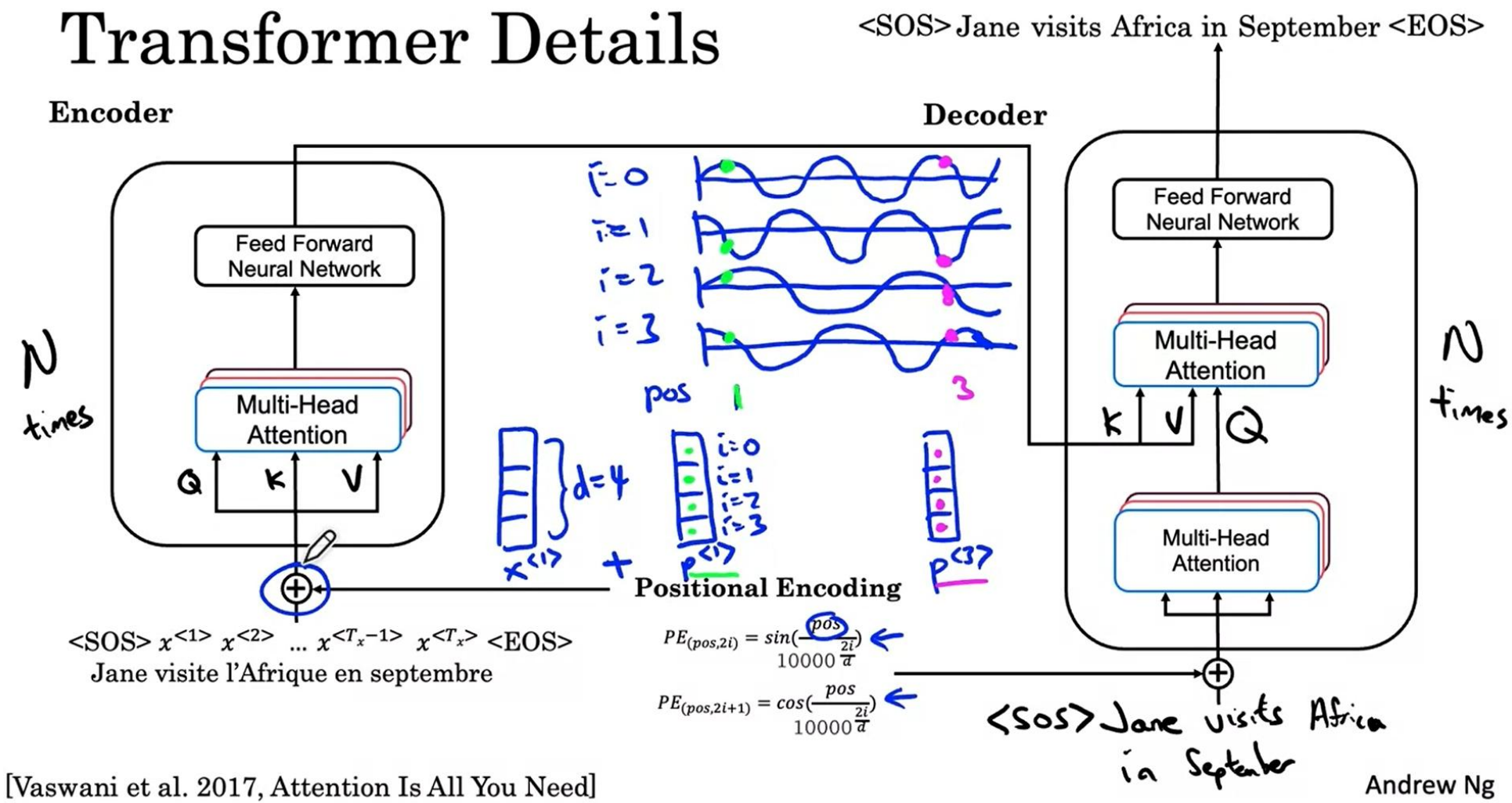
Andrew Ng

Transformer Details



[Vaswani et al. 2017, Attention Is All You Need]

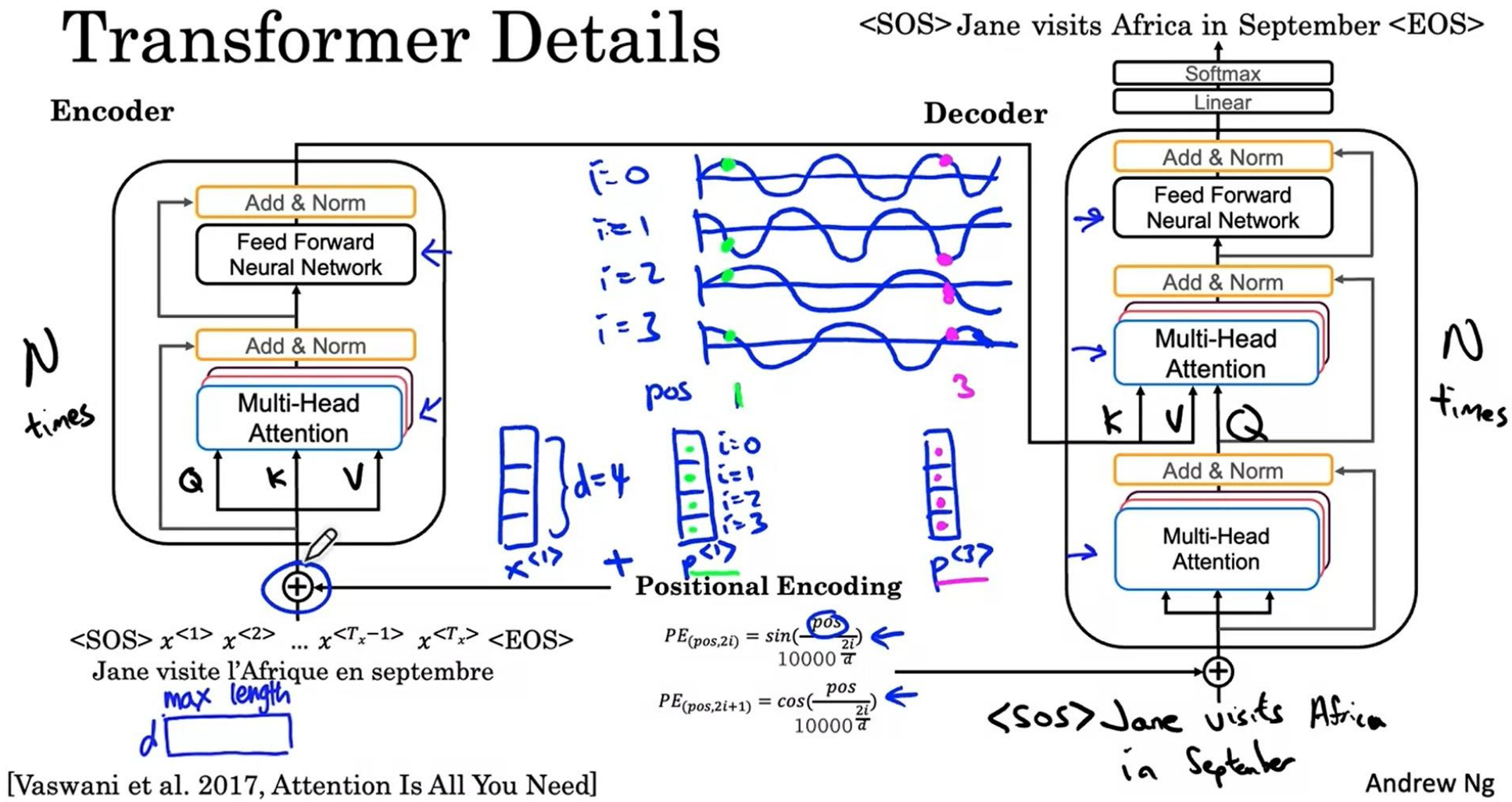
Transformer Details



[Vaswani et al. 2017, Attention Is All You Need]

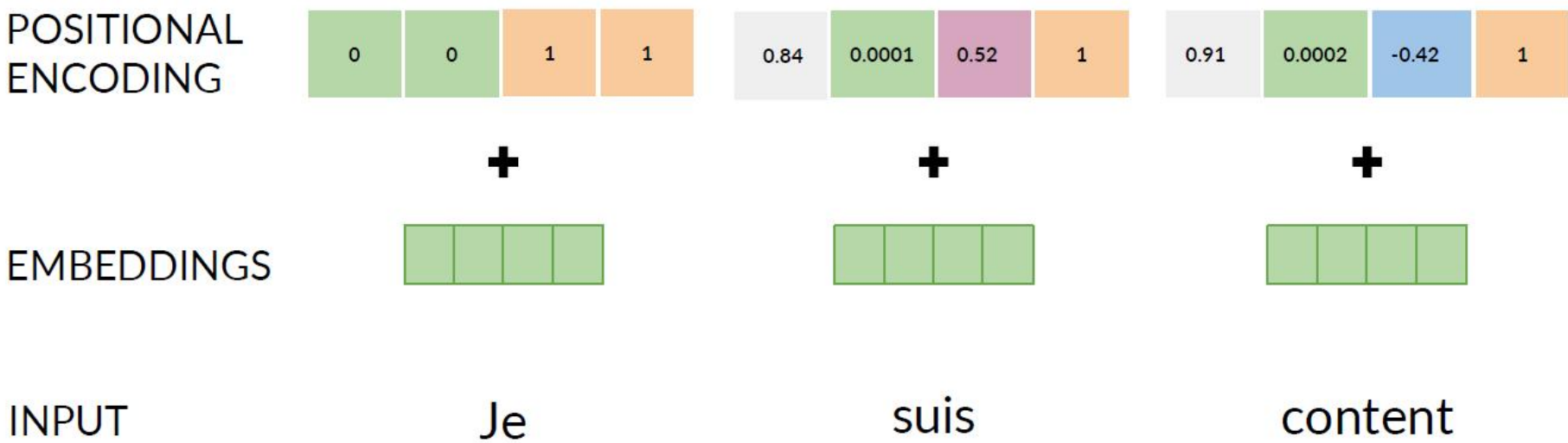
Andrew Ng

Transformer Details

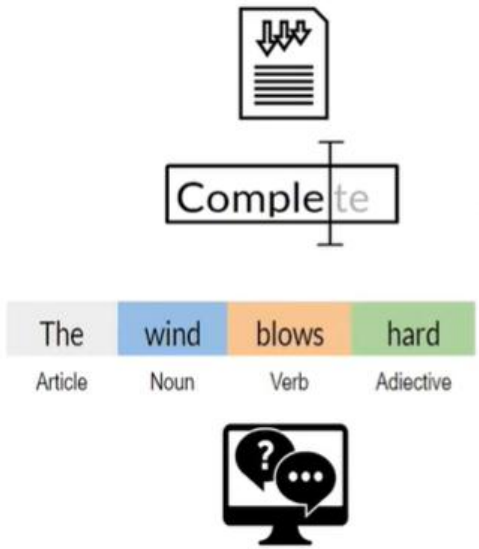


[Vaswani et al. 2017, Attention Is All You Need]

Andrew Ng


POSITIONAL ENCODING
0
0
1
1
0.84
0.0001
0.52
1
0.91
0.0002
-0.42
1
EMBEDDINGS
INPUT
Je
suis
content

Transformers'ların aktif kullanım alanları



Text summarization

Auto-Complete

Named entity recognition (NER)

Question answering (Q&A)

Translation

Chat-bots

Other NLP tasks

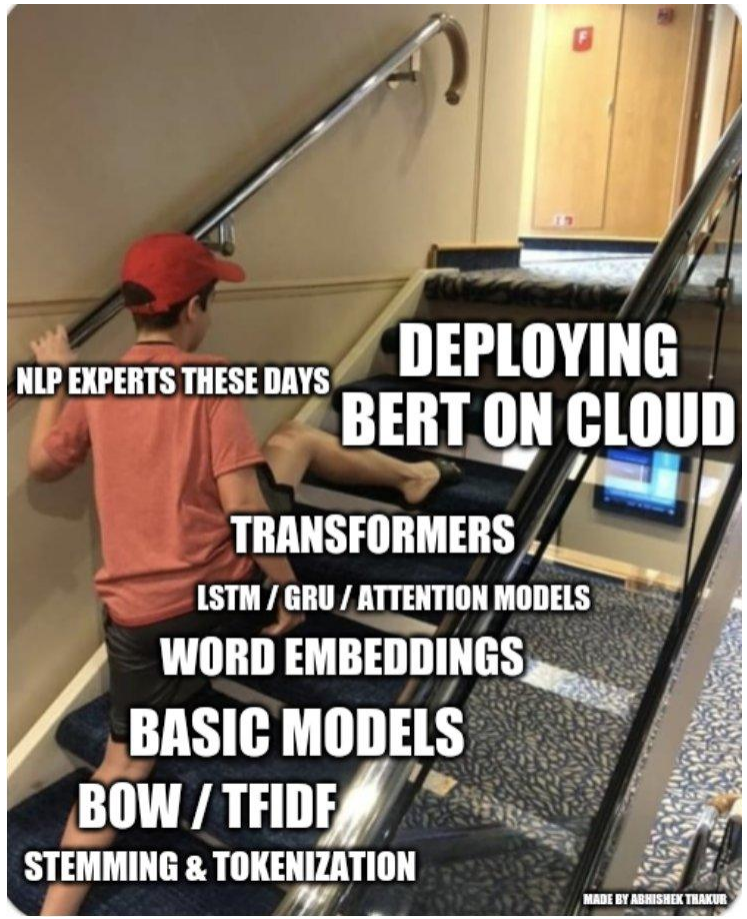
- Sentiment Analysis
- Market Intelligence
- Text Classification
- Character Recognition
- Spell Checking

- Önemli: BERT gibi dil modelleri Transformers'ları olduğu gibi kullanmazlar.

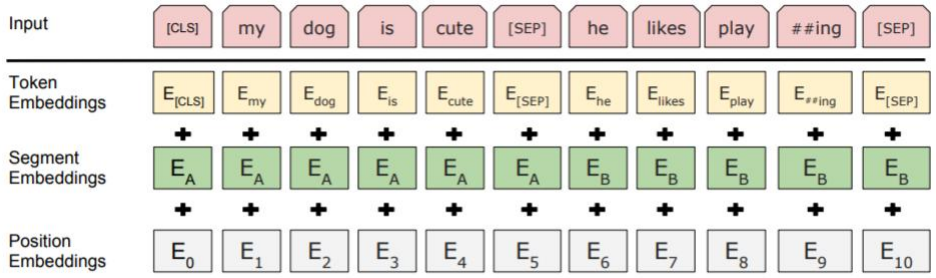
BERT Nedir?

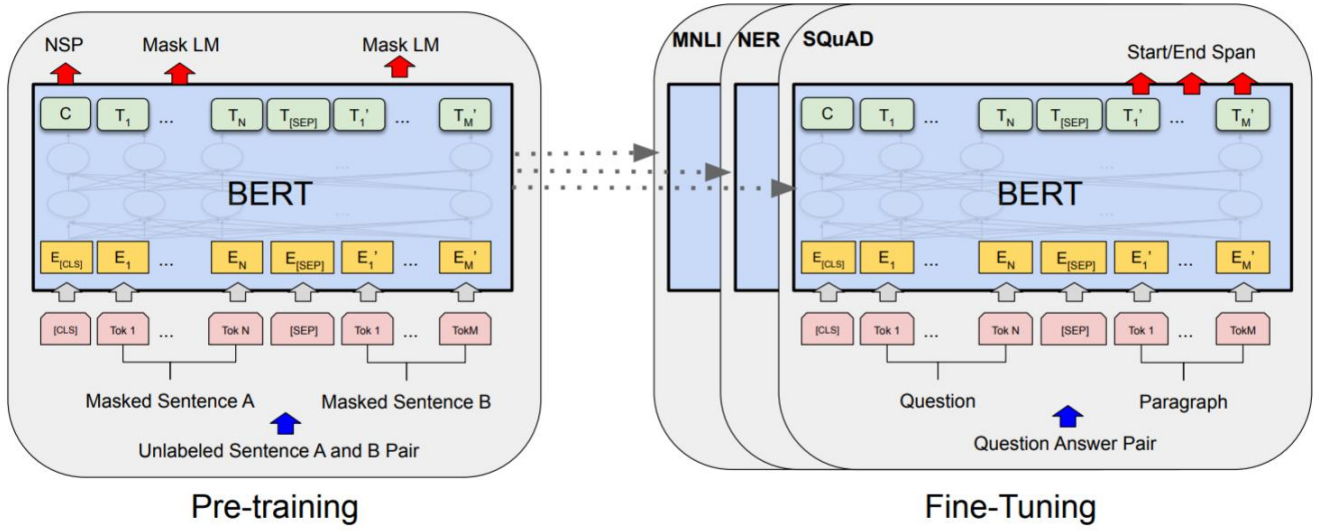
BERT (Bidirectional Transformers for Language Understanding), Google AI Language araştırmacıları tarafından yayınlanan görece yeni bir makaledir (2018). Çeşitli NLP görevlerinde state-of-the-art (en iyi ve en gelişmiş) sonuçları sunmaları, derin öğrenme topluluğu üzerinde önemli bir etki yarattı. Ayrıca BERT, kelime ve cümle düzeyinde bağlamsal bilgileri yakalamak için eğitim sürecinde maskeli dil modelleme yöntemini (Masked-language Modeling) ve sonraki cümleyi tahmin (Next Sentence Prediction) görevini kullanır.

- İşleri daha net hale getirmek için, bu noktada BERT yazarlarının fine-tuning ve özel görev eğitimi için kullandıkları özel belirteçleri anlamak önemlidir.



Bu belirteçler şunlardır: [CLS], [SEP] ve [MASK].
Kısaca, [CLS] her dizinin ilk simgesidir. [SEP], dizi çifti görevinin ön eğitiminde kullanılan dizi sınırlayıcı simgesidir. [MASK], maskelenmiş kelimeler için kullanılan belirteçtir. Sadece ön eğitim için kullanılır.
Girdi katmanı (Input), özel belirteçlerle birlikte dizi belirteçlerinin vektörüdür. Token Embeddings, belirteçlerin her biri için sözcüksel kimliklerdir. Segment Embeddings, A ve B cümlelerini ayırt etmek için sayısal bir sınıftır. Son olarak, Position Embeddings, dizideki her bir kelimenin konumunu ifade eder.





BERT, çok çeşitli dil görevlerinde kullanılabilir:

- Bir film hakkındaki yorumların ne kadar olumlu ya da olumsuz olduğunu belirleyebilir. (Duygu Analizi)
- Sohbet robotlarının sorularınızı yanıtlamasına yardımcı olur. (Soru cevaplama)
- Bir e-posta yazarken metninizi tahmin eder (Gmail). (Metin tahmini)
- Sadece birkaç cümle girişi ile herhangi bir konu hakkında bir makale yazabilir. (Metin oluşturma)
- Uzun yasal sözleşmeleri hızlı bir şekilde özetleyebilir. (Özetleme)
- Birden çok anlamı olan sözcükleri (“kız” gibi) çevreleyen metne göre ayırt edebilir. (Çok anlamlılık kararı)

BERT, Wikipedia (~2.5 milyar kelime) ve Google BooksCorpus (~800 milyon kelime) konusunda özel olarak eğitildi. Bu büyük bilgi veri kümeleri, BERT'in yalnızca İngilizce diliyle ilgili değil, aynı zamanda dünyamızla ilgili derin bilgisine de katkıda bulundu.

Bu kadar büyük bir veri kümesi üzerinde eğitim uzun zaman alır. BERT'nin eğitimi, yeni Transformer mimarisi sayesinde mümkün oldu ve TPU'lar (Tensor Processing Units - Google'ın özellikle büyük makine öğrenimi modelleri için oluşturulmuş özel devresi) kullanılarak hızlandırıldı. —64 TPU, BERT'i 4 gün boyunca eğitti.

Not: BERT'i daha küçük hesaplama ortamlarında (cep telefonları ve kişisel bilgisayarlar gibi) kullanmak için daha küçük BERT modellerine olan talep artmaktadır. Mart 2020'de 23 daha küçük BERT modeli piyasaya sürüldü. DistilBERT, BERT'nin daha hafif bir versiyonunu sunuyor; BERT performansının %95'inden fazlasını korurken %60 daha hızlı çalışır.

	Transformer Layers	Hidden Size	Attention Heads	Parameters	Processing	Length of Training
BERTbase	12	768	12	110M	4 TPUs	4 days
BERTlarge	24	1024	16	340M	16 TPUs	4 days

BERT's Performance on GLUE:

Task	Average	Grammatical	Sentiment Analysis	Similarity	Paraphrase	Question Similarity	Contradiction	Answerable	Entail
BERTLARGE	**82.1**	**60.5**	**94.9**	**86.5**	**89.3**	**72.1**	**86.7/85.9**	**92.7**	**70.1**
BERTBASE	79.6	52.1	93.5	85.8	88.9	71.2	84.6/83.4	90.5	66.4
OpenAI GPT	75.1	45.4	91.3	80.0	82.3	70.3	82.1/81.4	87.4	56.0
Pre-OpenAI SOTA	74.0	35.0	93.2	81.0	86.0	66.1	80.6/80.1	82.3	61.7
BiLSTM+ELMo+Attn	71.0	36.0	90.4	73.3	84.9	64.8	76.4/76.1	79.8	56.8

Maskeli Dil Modeli (Masked-language model) nedir?

MLM, bir cümledeki bir kelimeyi maskeleyerek (gizleyerek) ve BERT'i, maskelenen kelimeyi tahmin etmek için kapsanan kelimenin her iki tarafındaki kelimeleri çift yönlü olarak kullanmaya zorlayarak metinden çift yönlü öğrenmeyi etkinleştirir. Bu daha önce hiç yapılmamıştı.

Maskeli Dil Modelini bir örnek ile inceleyelim:

- Yolda bir arkadaşınızla konuşurken evinize vardığınızı ve asansöre bindiğinizi hayal edin. Çağrı tamamen kesilmeden önce duyduğunuz son şey:

  Haydarberk: “Yav, bu nlp dersinin haftaya [boşluk-cızırtı] var mıydı?”

Arkadaşınızın ne dediğini tahmin edebilir misiniz?

Bağlam ipuçları olarak (nlp dersiyle alakalı ne tür sorular olabileceğine dair) eksik kelimeden önceki ve sonraki kelimeleri çift yönlü olarak değerlendirerek, doğal olarak eksik kelimeyi tahmin edebilirsiniz. Arkadaşınızın 'ödevi', 'sınavı' ya da 'projesi' dediğini mi tahmin ettiniz?

Not: Biz insanlar bile bu durumların bazılarında hata yapmaya meyilliyiz. Bu nedenle, sık sık bir dil modelinin performans puanlarıyla "İnsan Performansı" karşılaştırması görebilirsiniz. Ve evet, BERT gibi daha yeni modeller insanlardan daha doğru tahminlerde bulunabilirler.

Önceki slaytta [boş] kelimeyi doldurmak için uyguladığınız çift yönlü metodoloji, BERT'nin en son teknolojiye sahip doğruluğa nasıl ulaştığına benzer. Belirlenmiş sözcüklerin rastgele %15'i eğitim sırasında gizlenir ve BERT'in görevi gizli sözcükleri doğru bir şekilde tahmin etmektir. Böylece modele doğrudan dil (ve kullandığımız kelimeler) hakkında bilgiler öğretilir.

Sonraki Cümle Tahmini Nedir? (Next Sentence Prediction) nedir?

NSP (Sonraki Cümle Tahmini), belirli bir cümlenin bir önceki cümleyi takip edip etmediğini tahmin ederek BERT'nin cümleler arasındaki ilişkileri öğrenmesine yardımcı olmak için kullanılır.

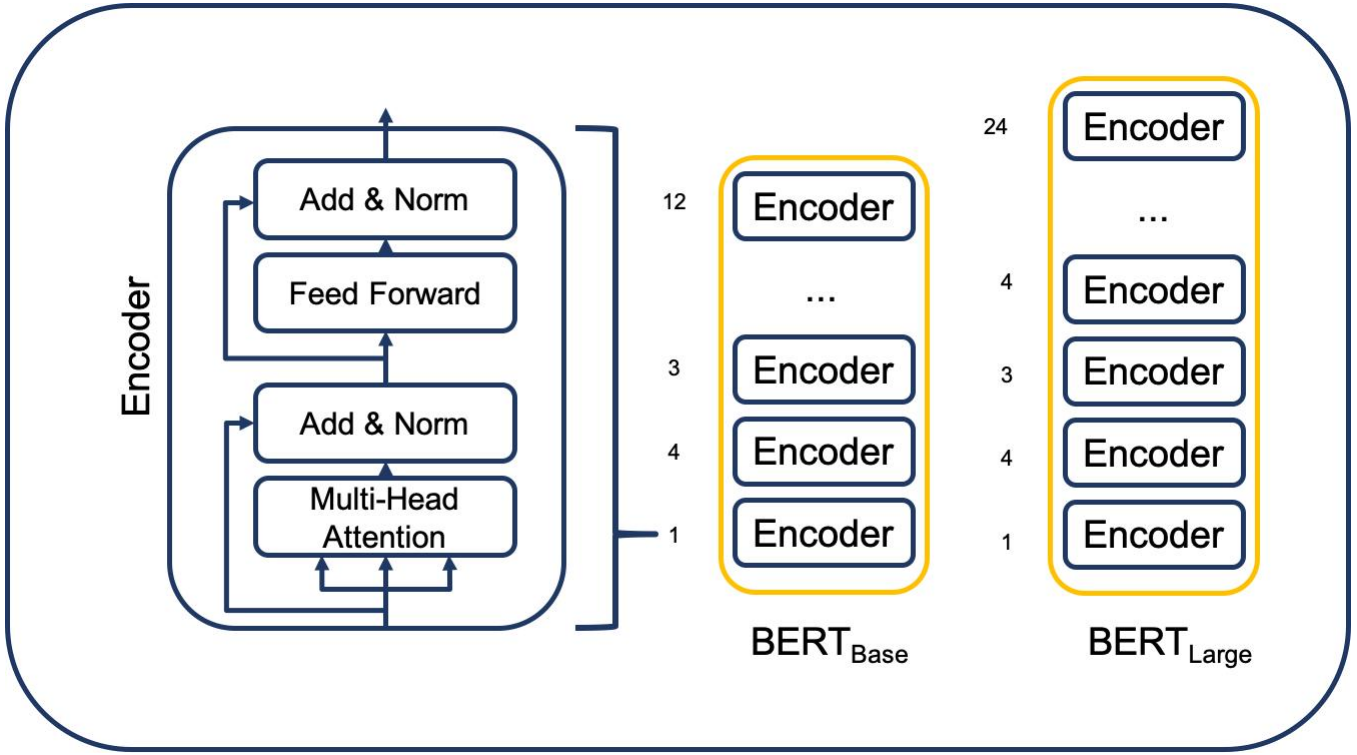
Sonraki Cümle Tahmin Örneği:

1) Ayşenaz alışverişe gitti. Yeni bir gömlek aldı. (doğru cümle çifti)
2) İsmail kahve yaptı. Vanilyalı dondurma külahları satılıktır. (yanlış cümle çifti)

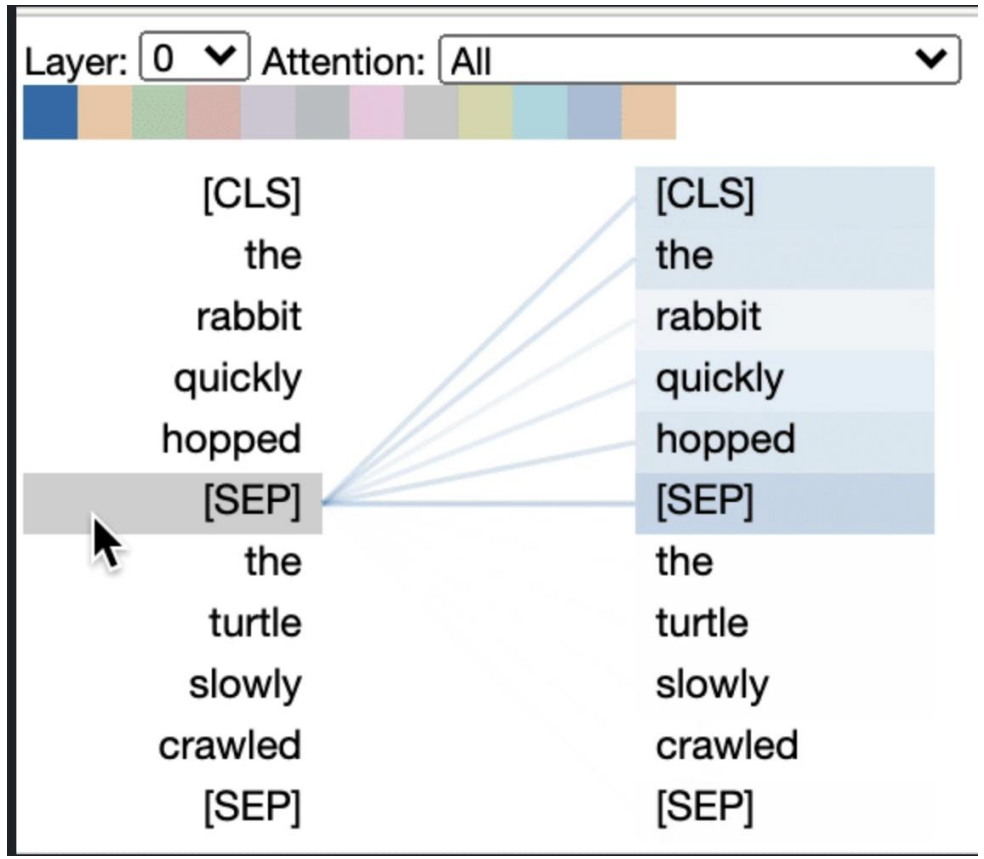
* Eğitimde, BERT'nin bir sonraki cümle tahmin doğruluğunu artırmasına yardımcı olmak için %50 doğru cümle çiftleri %50 rastgele cümle çiftleriyle karıştırılır.

- BERT, aynı anda hem MLM (%50) hem de NSP (%50) konusunda eğitilmiştir.

BERT modeli Decoder içermez.



BertViz (https://github.com/jessevig/bertviz)



Güncel NLP’de derin öğrenme tabanlı ilginç ve farklı tartışmalar/çalışmalar

> *I did enjoy [your essay], but have my usual qualms. Take GPT-3 – I’m sure you saw the lead article in the NYT magazine collapsing in awe about its ability to mimic some regularities in data. In fact, its only achievement is to use up a lot of California’s energy. You can’t go to a physics conference and say: I’ve got a great theory. It accounts for everything and is so simple it can be captured in two words: “Anything goes.”*
>
> *All known and unknown laws of nature are accommodated, no failures. Of course, everything impossible is accommodated also.*
>
> *That’s GPT-3. Works as well or better for 45 terabytes of data from impossible languages.*
>
> *It’s been understood forever that a theory has to answer two kinds of questions: Why this? Why not that?*
>
> *Noam*

Noam Chomsky’nin GPT-3 hakkındaki yorumu

DeepMind

Ethical and social risks of harm from Language Models

Laura Weidinger[1], John Mellor[1], Maribeth Rauh[1], Conor Griffin[1], Jonathan Uesato[1], Po-Sen Huang[1], Myra Cheng[1,2], Mia Glaese[1], Borja Balle[1], Atoosa Kasirzadeh[1,3], Zac Kenton[1], Sasha Brown[1], Will Hawkins[1], Tom Stepleton[1], Courtney Biles[1], Abeba Birhane[1,4], Julia Haas[1], Laura Rimell[1], Lisa Anne Hendricks[1], William Isaac[1], Sean Legassick[1], Geoffrey Irving[1] and Iason Gabriel[1]

[1]DeepMind, [2]California Institute of Technology, [3]University of Toronto, [4]University College Dublin

Abstract

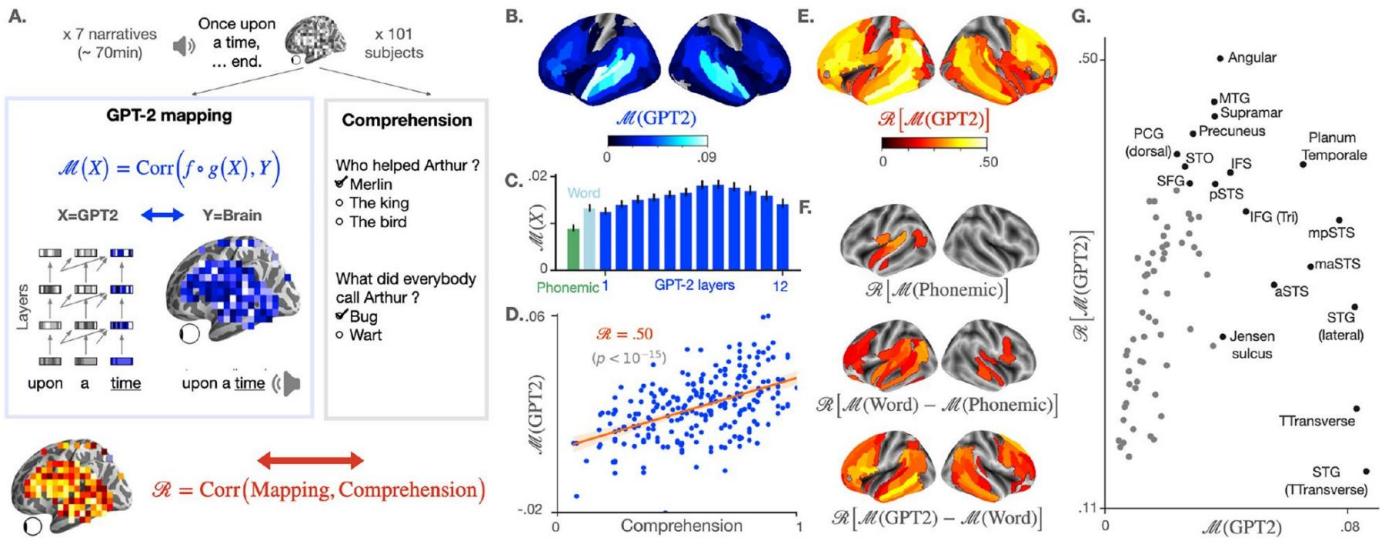
This paper aims to help structure the risk landscape associated with large-scale Language Models (LMs). In order to foster advances in responsible innovation, an in-depth understanding of the potential risks posed by these models is needed. A wide range of established and anticipated risks are analysed in detail, drawing on multidisciplinary literature from computer science, linguistics, and social sciences.

The paper outlines six specific risk areas: I. Discrimination, Exclusion and Toxicity, II. Information Hazards, III. Misinformation Harms, IV. Malicious Uses, V. Human-Computer Interaction Harms, VI. Automation, Access, and Environmental Harms.

Deep language algorithms predict semantic comprehension from brain activity

Charlotte Caucheteux[1,2]✉, Alexandre Gramfort[2] & Jean-Rémi King[1,3]

Deep language algorithms, like GPT-2, have demonstrated remarkable abilities to process text, and now constitute the backbone of automatic translation, summarization and dialogue. However, whether these models encode information that relates to human comprehension still remains controversial. Here, we show that the representations of GPT-2 not only map onto the brain responses to spoken stories, but they also predict the extent to which subjects understand the corresponding narratives. To this end, we analyze 101 subjects recorded with functional Magnetic Resonance Imaging while listening to 70 min of short stories. We then fit a linear mapping model to predict brain activity from GPT-2's activations. Finally, we show that this mapping reliably correlates ($\mathcal{R} = 0.50, p < 10^{-15}$) with subjects' comprehension scores as assessed for each story. This effect peaks in the angular, medial temporal and supra-marginal gyri, and is best accounted for by the long-distance dependencies generated in the deep layers of GPT-2. Overall, this study shows how deep language models help clarify the brain computations underlying language comprehension.



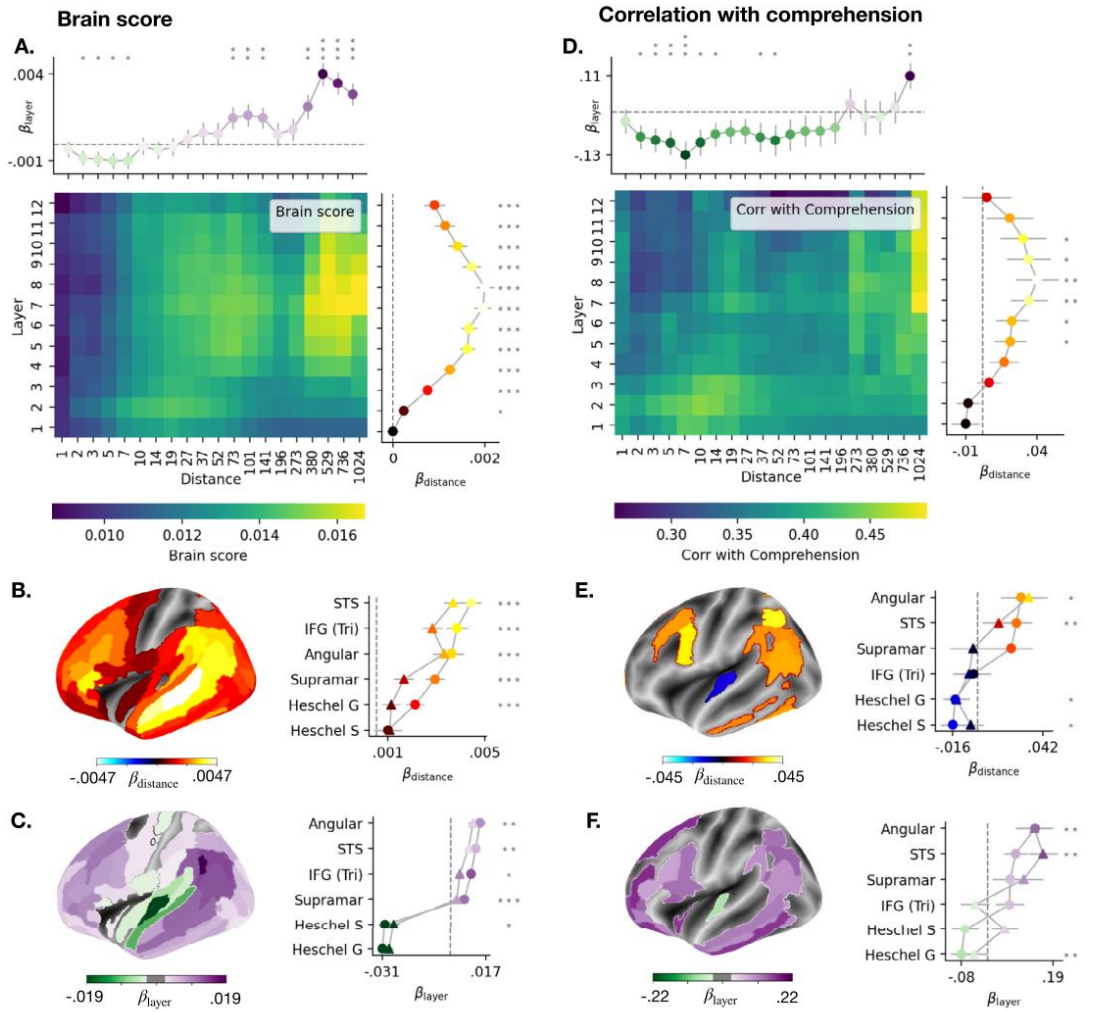


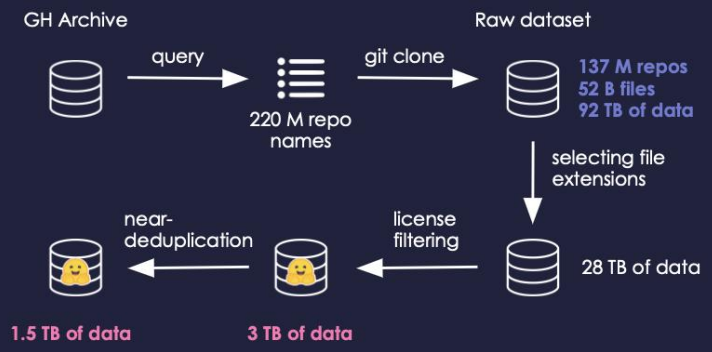
Figure 2. Impact of GPT-2's attention span on brain scores and comprehension scores. (**A**) The heatmap displays the average (across subjects, stories and voxels) brain scores as a function of attention span ("distance") and layers. The top line displays the layer coefficients for each attention span (averaged across subjects, stories and voxels). The right line displays the distance coefficient for each layer (averaged across subjects, stories and voxels). The error bars correspond to the Standard Errors of the Mean (SEM) across subject-story pairs. (**B**) Distance coefficients for each brain region (averaged across subjects and stories). Statistical significance is assessed with a Wilcoxon test across subject-story pairs. (**C**) Layer coefficients for each brain region (averaged across subjects and stories). (**D**)–(**F**) Similar as (**A**)–(**C**), but the layer (and distance, respectively) coefficients now assess the relationship between layer (or distance, respectively) and comprehension scores. Statistical significance is assessed using a bootstrapping procedure with 1000 subsamples of subject-story pairs. Error bars are standard deviation across subsamples. For all brain maps, only significant values are displayed ($p < 0.05$ after FDR correction across brain regions).

@BigCodeProject
https://www.bigcode-project.org/
contact@bigcode-project.org

The Stack

3 TB of permissive code data

Dataset Collection



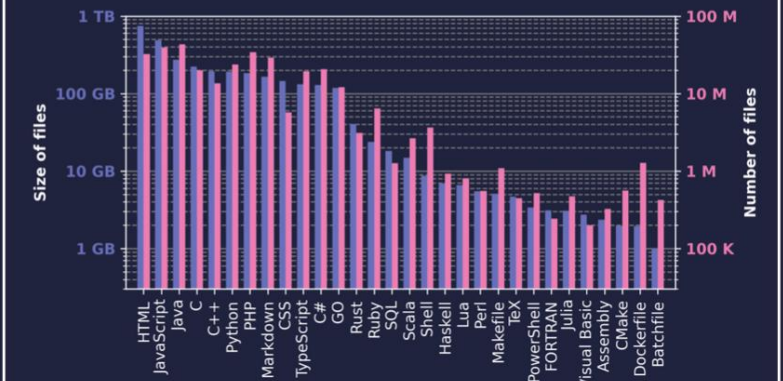
Find the filtered and deduplicated datasets at: www.hf.co/bigcode

Evaluation

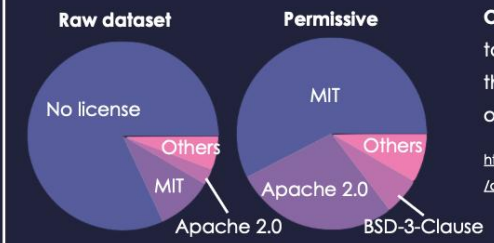
We trained several GPT-2 models (350M parameters) on different parts of the dataset both with and without near-deduplication. The models trained on the Python subset of The Stack performed on par with CodeX and CodeGen of similar size when using near-deduplication. Since HumanEval data leaked into the training dataset we also trained decontaminated models

Dataset	Filtering	pass@1	pass@10	pass@100
Codex (300M)	unknown	13.17	20.17	36.27
CodeGen (350M)	unknown	12.76	23.11	35.19
Python all-license	None	13.11	21.77	36.67
	Near-dedup	16.60	27.82	44.00
	+decontaminate	17.34	27.64	45.52
Python permissive-license	None	10.99	15.94	27.21
	Near-dedup	13.94	22.36	37.00
	+decontaminate	12.89	22.26	36.01

Programming Languages



Licensing + Governance



Opt-out: If users would like to exclude their code from the corpus we have an opt-out mechanism. Visit:

https://www.bigcode-project.org/docs/about/the-stack/

All licenses used to filter the dataset:
MIT-0 | MIT | MIT-feh | Apache-2.0 | BSD-3-Clause | BSD-3-Clause-Clear BSD-3-Clause-No-Nuclear-License-2014 | BSD-2-Clause | CC0-1.0 | EPL-1.0 MPL-2.0 | Unlicense | ISC | Artistic-2.0 | deprecated_LGPL-3.0+ deprecated_LGPL-2.1+ | ECL-2.0 | SHL-0.51 | MPL-2.0-no-copyleft-exception

Usage

```
!pip install datasets
from datasets import load_dataset

# full dataset (3TB of data)
ds = load_dataset("bigcode/the-stack", split="train")

# near-deduplicated dataset (1.5TB of data)
ds = load_dataset("bigcode/the-stack-dedup", split="train")

# specific language (e.g. Dockerfiles)
ds = load_dataset("bigcode/the-stack", data_dir="data/dockerfile", split="train")

# dataset streaming (will only download the data as needed)
ds = load_dataset("bigcode/the-stack", streaming=True, split="train")
for sample in iter(ds): print(sample["content"])
```

For more info visit about the Datasets library visit: hf.co/docs/datasets

Dataset Trivia

- There are 1,467,018 files containing **"hello world"** in the dataset.
- The message **"Your mom ate the database"** (in German) appears in a file.
- It would take a single person **32,800 years** to type all the code from scratch.
- The Stack is **150x larger** than the English Wikipedia.

Printed double sided on A4 paper the stacked pile would almost reach **11x the Mount everest** and you could cover **8x the distance between earth and moon** if the paper is aligned side-by-side.

Human-level play in the game of *Diplomacy* by combining language models with strategic reasoning

Meta Fundamental AI Research Diplomacy Team (FAIR)†, Anton Bakhtin[1]‡, Noam Brown[1]*‡, Emily Dinan[1]*‡, Gabriele Farina[1], Colin Flaherty[1]‡, Daniel Fried[1,2], Andrew Goff[1], Jonathan Gray[1]‡, Hengyuan Hu[1,3]‡, Athul Paul Jacob[1,4]‡, Mojtaba Komeili[1], Karthik Konath[1], Minae Kwon[1,3], Adam Lerer[1]*‡, Mike Lewis[1]*‡, Alexander H. Miller[1]‡, Sasha Mitts[1], Adithya Renduchintala[1]‡, Stephen Roller[1], Dirk Rowe[1], Weiyan Shi[1,5]‡, Joe Spisak[1], Alexander Wei[1,6], David Wu[1]‡, Hugh Zhang[1,7]‡, Markus Zijlstra[1]

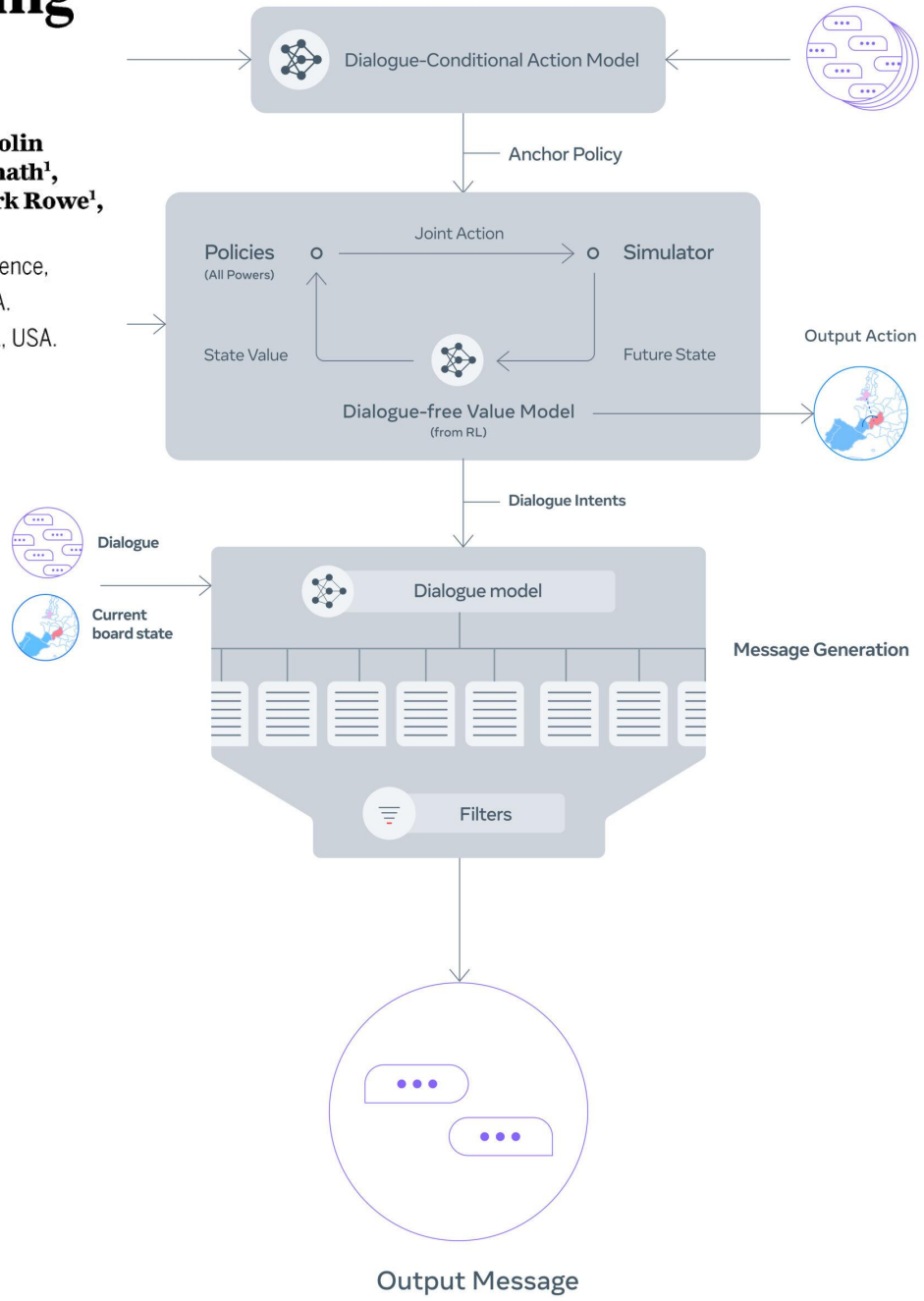
[1]Meta AI, 1 Hacker Way, Menlo Park, CA, USA. [2]Language Technologies Institute, Carnegie Mellon University, Pittsburgh, PA, USA. [3]Department of Computer Science, Stanford University, Stanford, CA, USA. [4]Computer Science and Artificial Intelligence Laboratory, Massachussetts Insititute of Technology, Cambridge, MA, USA. [5]Department of Computer Science, Columbia University, New York, NY, USA. [6]Department of Computer Science, University of California, Berkeley, Berkeley, CA, USA. [7]EconCS Grooup, Harvard University, Cambridge, MA, USA.

"An agent that can play at the level of humans in a game as strategically complex as Diplomacy is a true breakthrough for cooperative AI."

Yann LeCun
VP & Chief AI Scientist, Meta AI

By building CICERO, Meta AI has created the first AI agent to achieve human-level performance in the complex natural language strategy game Diplomacy*. CICERO demonstrated this by playing with humans on webDiplomacy.net, an online version of the game, where CICERO achieved more than double the average score of the human players and ranked in the top 10% of participants who played more than one game.

This breakthrough rests in the achievement of combining two different areas of AI: **strategic reasoning and natural language processing**. The integration of these techniques gives CICERO the ability to reason and strategize with regard to players' motivations, then use natural language to communicate, reach agreements to achieve shared objectives, form alliances and coordinate plans.



https://ai.facebook.com/research/cicero/

Galactica: A Large Language Model for Science

Abstract

Information overload is a major obstacle to scientific progress. The explosive growth in scientific literature and data has made it ever harder to discover useful insights in a large mass of information. Today scientific knowledge is accessed through search engines, but they are unable to organize scientific knowledge alone. In this paper we introduce Galactica: a large language model that can store, combine and reason about scientific knowledge. We train on a large scientific corpus of papers, reference material, knowledge bases and many other sources. We outperform existing models on a range of scientific tasks. On technical knowledge probes such as LaTeX equations, Galactica outperforms the latest GPT-3 by 68.2% versus 49.0%. Galactica also performs well on reasoning, outperforming Chinchilla on mathematical MMLU by 41.3% to 35.7%, and PaLM 540B on MATH with a score of 20.4% versus 8.8%. It also sets a new state-of-the-art on downstream tasks such as PubMedQA and MedMCQA dev of 77.6% and 52.9%. And despite not being trained on a general corpus, Galactica outperforms BLOOM and OPT-175B on BIG-bench. We believe these results demonstrate the potential for language models as a new interface for science. We open source the model for the benefit of the scientific community[1].

Prompt

Sulfuric acid reacts with sodium chloride, and gives _____ and _____:

```
\[ \ce{ NaCl + H2SO4 ->
```

Generated Answer

$$\mathrm{NaCl} + \mathrm{H_2SO_4} \longrightarrow \mathrm{NaHSO_4} + \mathrm{HCl}$$

Figure 10: Chemical Reactions. We prompt based on a description and reactants, and evaluate whether the generated products are correct.

Task: Convert the SMILES to IUPAC Name

Example: `CC(C)(C)C(=O)N(CC1=NC(=CS1)C(=O)OC)C2CCCCC2`

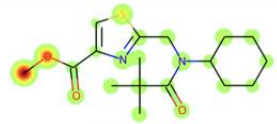
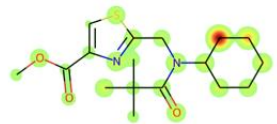
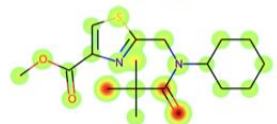
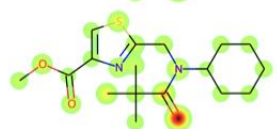
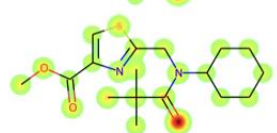
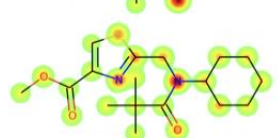
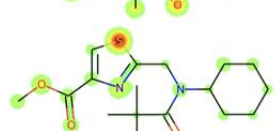
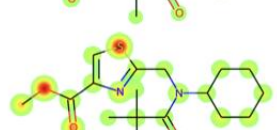
Atomic Attention	Predicted So Far	Token Predicted
	-	`methyl`
	methyl 2-[[cyclohexyl	`cyclohexyl`
	methyl 2-[[cyclohexyl-(2,2-	`dimethyl`
	methyl 2-[[cyclohexyl-(2,2-dimethyl	`prop`
	methyl 2-[[cyclohexyl-(2,2-dimethylprop	`anoyl`
	methyl 2-[[cyclohexyl-(2,2-dimethylpropanoyl)	`amino`
	methyl 2-[[cyclohexyl-(2,2-dimethylpropanoyl)]amino] methyl]	`th`
	methyl 2-[[cyclohexyl-(2,2-dimethylpropanoyl)]amino] methyl]th	`iazole`
	methyl 2-[[cyclohexyl-(2,2-dimethylpropanoyl)]amino] methyl]thiazole-4-	`carboxylate`

Figure 13: Attending to Functional Groups. Galactica uses its knowledge of chemistry to help with the IUPAC Naming task. At each stage of prediction, it attends to the part of the molecular graph associated with the group name, e.g. for "amino" it attends to the nitrogen atom; for thiazole, the sulphur atom.

Derin öğrenme çalışmaları için platformlar

- https://huggingface.co
- https://www.kaggle.com
- https://paperswithcode.com
- https://colab.research.google.com

Konulara ve güncel araştırmalara daha kapsamlı bakmak isteyenler için kaynaklar

Tarihsel Yayınlardan Bazıları

- - (1989) Handwritten Digit Recognition with a Back-Propagation Network: https://lnkd.in/gtv_Q7iv
- - (1997) LONG SHORT-TERM MEMORY: https://lnkd.in/gCWJjxJv
- - (2010) Understanding the difficulty of training deep feedforward neural networks: https://lnkd.in/gPSak5R3
- - (2011) ReLU activation function (Deep Sparse Rectifier Neural Networks): https://lnkd.in/g8kgSfjT
- - (2012) ADADELTA: AN ADAPTIVE LEARNING RATE METHOD: https://lnkd.in/gTMFGZ6J
- - (2012) Random Search for Hyper-Parameter Optimization: https://lnkd.in/gtx_ABwi
- - (2012) Improving neural networks by preventing co-adaptation of feature detectors: https://lnkd.in/g49Sp6HE
- - (2012) Deep Learning of Representations for Unsupervised and Transfer Learning: https://lnkd.in/g9yWA86k
- - (2013) Efficient Estimation of Word Representations in Vector Space: https://lnkd.in/gC62AchR
- - (2013) Maxout Networks: https://lnkd.in/gC_KvJjT
- - (2014) Dropout: A Simple Way to Prevent Neural Networks from Overfitting: https://jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf
- - (2014) On the Properties of Neural Machine Translation: Encoder–Decoder Approaches: https://lnkd.in/g-rRQ6km
- - (2014) ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION: https://lnkd.in/gtsh7SGQ
- - (2014) GloVe: Global Vectors for Word Representation: https://lnkd.in/gA8bnnX2
- - (2015) Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift: https://lnkd.in/gmptQTXY
- - (2015) Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification: https://lnkd.in/gCUy7fzR
- - (2016) Layer Normalization: https://lnkd.in/gTad4iHE
- - (2017) Attention Is All You Need: https://lnkd.in/gUts7Sjq

Oldukça kapsamlı olan kaynaklar:

- Speech and Language Processing: https://web.stanford.edu/~jurafsky/slp3/
- (2021) Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network: https://arxiv.org/pdf/1808.03314.pdf