# Bağlam Farkında Sinirsel Dil Modelleri

Mehmet Fatih AMASYALI
Hesaplamalı Anlambilim
Ders Notları
BÖLÜM 5

Yıldız Teknik Üniversitesi
Bilgisayar Mühendisliği Bölümü





## İçerik

- Bağlam bağımsız dil modelleri (gördük)
- Dağıtık Temsil
- RNN, GRU, LSTM, attention, transformer
- Self-supervised learning
  - Kapalı kelime tahmini
  - Sonraki cümle tahmini
  - Başka?
- Byte pair encoding
- Bağlam bağımlı dil modelleri
  - ULMfit
  - BERT
  - GPT
  - Albert
  - Roberta
  - BART
  - Vb.

# Dağıtık Temsil
# Distributed representations*

- Diyelim ki resimlerdeki insanları kadın/erkek, gözlüklü/gözlüksüz, uzun/kısa diye belirlemek istiyoruz. Eğer 8 sınıf tanımlarsak her sınıftan örneklere ihtiyacımız olur.
- Eğer 3 alt özelliği öğrenen 3 binary sınıflandırıcı tanımlarsak, eğitim kümemizde uzun, kadın, gözlüksüz bir örnek olmasa bile onu tanımlayabiliriz.
- d adet alt özellikle, 2 üzeri d adet farklı kombinasyonu ifade edebiliriz. Yani N adet sınıf, böyle örtüşmeli (non mutually exclusive) özellikler taşıyorsa $\log_2 N$ adet alt özellikle ifade edilebilir.
- [*] https://arxiv.org/pdf/1206.5538.pdf



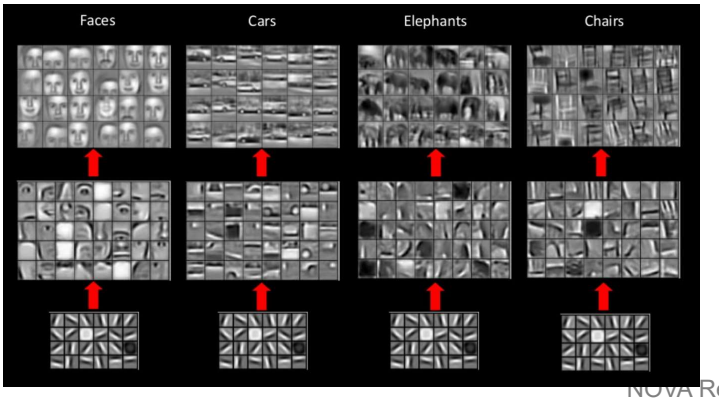


# Distributed representations

- Örneğin 1024 sınıf varsa. Klasik yaklaşımla 1024 prototip öğrenmemiz gerekir. Mesela d*d'lik resimleri sınıflandıracaksak ağın yapısı: (d*d)*1024 olur. d*d*1024 parametre öğrenmesi gerekir ağın.
- Ama 10 alt özellik kullanırsak ağın yapısı bir katman artıp (d*d)*10*1024 olur. Son katman alt özelliklerin kombinasyonlarını sınıf etiketlerine eşlemeyi öğrenir. (d*d)*10 + (10*1024) parametre öğrenir ağ.
- Karşılaştırırsak d=128 için klasik yaklaşım yaklaşık 4 milyon (2 üzeri 24), distributed yaklaşım yaklaşık 180 bin ☺
- Katman sayısını arttırarak öğrenmemiz gereken parametre sayısını azalttık ☺

# Distributed representations

- Peki bir sınıfı ifade eden, sınıflar arası ortak olabilecek parçaları (alt özellikleri) önceden nasıl bilebiliriz. Öğrenelim ☺
- Bir MLP'nin her bir gizli katmanındaki her bir nöron bir alt özelliği öğrenir.
- Bir CNN'deki her bir filtre bir alt özelliği öğrenir. Varsayım: Tüm resimler ortak küçük parçalardan oluşur. En alt seviyede bu basit özellikler sonrasında bunların kombinasyonları öğrenilir.
  https://www.slideshare.net/akshaymuroor/deep-learning-24650492







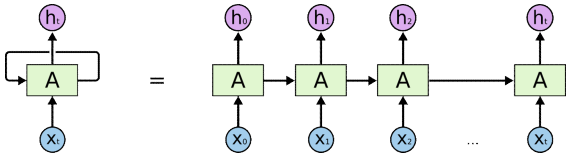

# Capsule Networks [*]

- Küçük parçaların resmin herhangi bir yerinde varlığı yeterlidir. Pooling katmanı nerede olduğunu önemsizleştirir. Sadece olup olmadığını söyler.
- Aslında bu parçaların birbirlerine göre konumları da önemlidir. Bir yüz resminde ağızla gözün yerini değiştirirseniz hala bir yüz olur mu? CNN'lere alternatif olan Capsule network'ler bu birbirine göre konumları da işe katar.

[*] https://medium.com/ai³-theory-practice-business/understanding-hintons-capsule-networks-part-i-intuition-b4b559d1159b

# Recurrent Neural Networks [*]

- Girişin sekans olduğu uygulamalar için
- Bir cümlenin kelimeleri, bir filmin frame leri, bir konuşmanın fonemleri vb.
- h(t): t. çıkış, x(t): t.giriş
- h(t), x(t) ve h(t-1)'e bağlı (aslında x0 … xt ye bağlı)



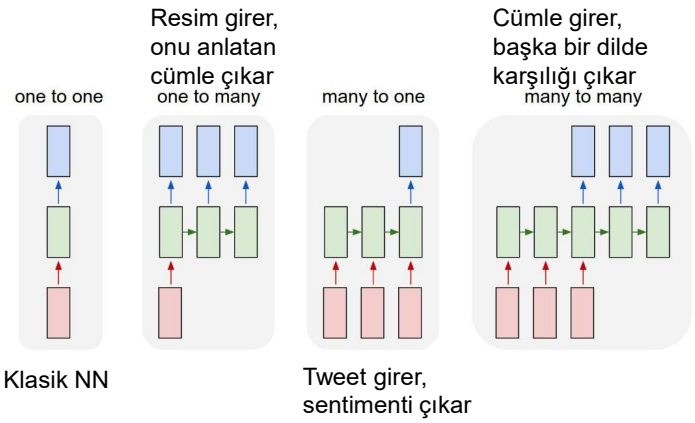

- Bu gösterim klasik ağların da sabit büyüklükte geçmişi işe katmalarının sağlayabileceğini gösterir. Giriş x yerine birleştirilmiş ($x_t$, $x_{t-1}$, … $x_{t-k}$) olarak verilir. Bu sayede $y_t$ nin sadece $x_t$ ye değil geçmişine de bağlı olarak modelleyebiliriz. Son zamanlarda RNN lerin işini yapan CNN ler öneriliyor.

[*] http://colah.github.io/posts/2015-08-Understanding-LSTMs/





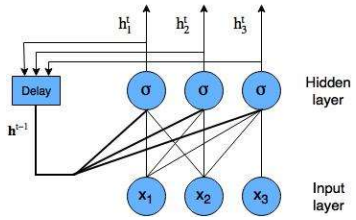
# RNN uygulama türleri [*]




[*] http://karpathy.github.io/2015/05/21/rnn-effectiveness/

# RNN'lerde saklı nöron sayısı [*]

- 3 saklı nöronlu bir RNN (3 boyutlu bir state tutar), yeni girişler geldikçe bu state'ler güncellenir. State ler, o ana kadar gelen girişlerin temsilidir.
- $h_t=\tanh(W_{hh}h_{t-1}+W_{xh}x_t)$
- Öğrenilen ağırlıklar: input ile hidden layer (state) arası ($W_{xh}$) ve hidden layerların t ile t-1 leri arası ($W_{hh}$)



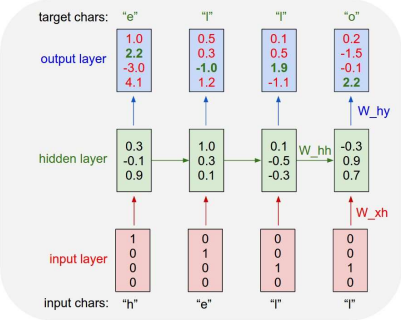

[*] http://adventuresinmachinelearning.com/recurrent-neural-networks-lstm-tutorial-tensorflow/





# RNN ile bir sonraki harfi tahmin

- input ve output 4 boyutlu (one-hat encoding, sözlük 4 karakter), 3 hidden nöron var, RNN'in üzerine bir katman daha eklenmiş. W_hy de öğreniliyor.




- Bir sonraki harf yerine bir sonraki kelimeyi tahmin etmek için one-hat kullansak? Alternatif?

# Daha fazla RNN

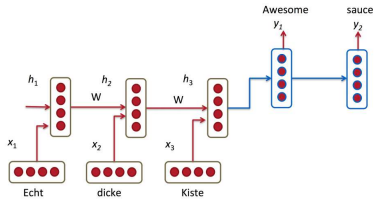
- RNN'lerin türevleri:
  - LSTM, BiLSTM, GRU, NTM, …
  - https://distill.pub/2016/augmented-rnns/
- RNN'lerin düşüşü
  - https://towardsdatascience.com/the-fall-of-rnn-lstm-2d1594c74ce0





# RNN'lerle Dil Çevirisi

ya da diziden diziye dönüşümler



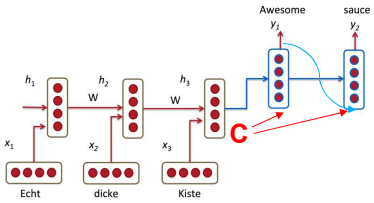

- En basit mimari
- Encoder-decoder
- Encoder:
- $h_t = \varphi(h_{t-1}, x_t) = f(W^{hh} h_{t-1} + W^{hx} x_t)$
- Decoder:
- $h_t = \varphi(h_{t-1}) = f(W^{hh} h_{t-1})$
- $y_t = softmax(W^s h_t)$

[*] https://www.youtube.com/watch?v=Keqep_PKrY8

[*] http://www.wildml.com/2016/01/attention-and-memory-in-deep-learning-and-nlp/

# RNN'lerle Dil Çevirisinde bazı iyileştirmeler

- Tüm cümle tek bir state'e (C) indirgenmiş
- Encoder: Cümleyi tersten verelim. Encode ve decode edilecek bilgiler arası mesafeyi azaltalım.
- Decoder:
- $h_t = \varphi(h_{t-1}, C, y_{t-1})$
- her biri için birer W var
- $y_t = softmax(W^s \begin{bmatrix} h_t \\ C \end{bmatrix})$

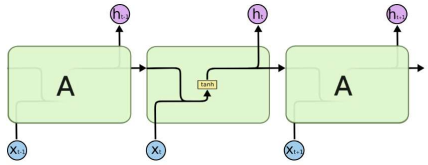



[*] https://www.youtube.com/watch?v=QuELiw8tbx8

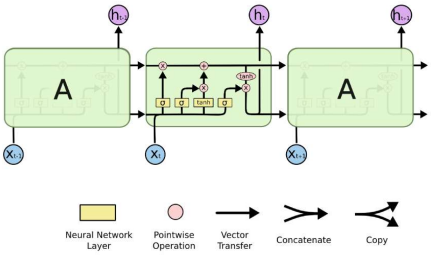




# LSTM



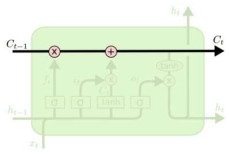

RNN: $h_{t-1}$ $ve$ $x_t$ birleştirilip tek bir W matrisi öğrenilir.
Bazı $x_t$ ler h'a etki etmeyebilir.
Bunu LSTM'ler yapabilir



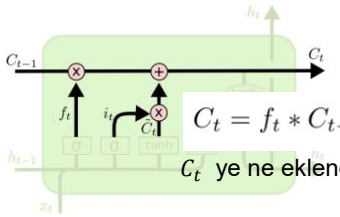

LSTM: 4 W matrisi (Wf, Wi, Wc, Wo ) öğrenilir. (Sarılar)

[*] https://colah.github.io/posts/2015-08-Understanding-LSTMs/
[*] https://distill.pub/2016/augmented-rnns/

# LSTM

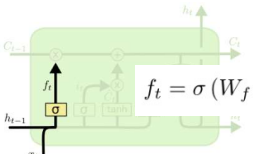


$C_{t-1}$ belli oranda unutulur/saklanır.
İlk X operatörü

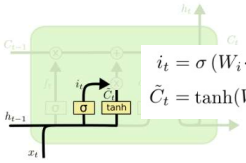


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

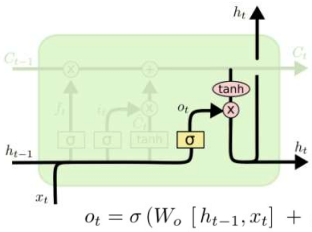
$C_t$ ye ne eklenecek?



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

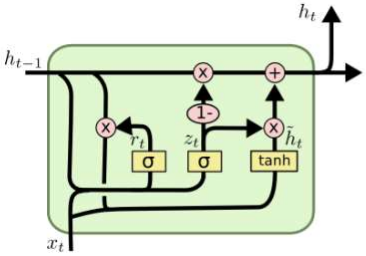


$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$





# GRU

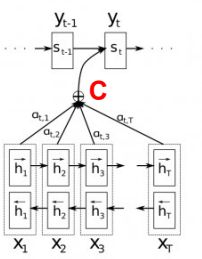


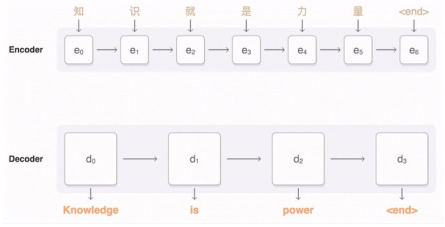
$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

LSTM'e göre daha basit.
3 W matris (Wz, Wr, W) öğrenilir. (Sarılar)

# Attention (Dikkat)

- Encoder'ın tüm state'leri etkin, ama ne ölçüde etkin (α) olacakları öğreniliyor




$\alpha^{i,j}$: i. çıkışa j. girişin ne kadar etkisi olacağı

$$\sum \alpha^{1,t} = 1$$

$$h^t = (h^{\rightarrow t}\ h^{\leftarrow t})$$

$$C_1 = \sum_t \alpha^{1,t}\ h^t$$

[*] http://www.wildml.com/2016/01/attention-and-memory-in-deep-learning-and-nlp/
[*] https://medium.com/syncedreview/history-and-frontier-of-the-neural-machine-translation-dc981d25422d
[*] https://www.youtube.com/watch?v=IxQtK2SjWWM&list=PL3FW7Lu3i5Jsnh1rnUwq_TcylNr7EkRe6&index=11
[*] https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html





# Transformers

- Attention is all you need
- RNN yok. Dolayısıyla paralelleştirilebilir
- Giriş ve çıkış dizisi arasındaki bağlılıkları klasik attention buluyor.
- Transformers girişin ve çıkışın kendi içindeki bağımlılıklarını da işe katıyor.
- Multihead attention

[*] https://jalammar.github.io/illustrated-transformer/
[*] https://mlexplained.com/2017/12/29/attention-is-all-you-need-explained/

# Bağlam Tabanlı Dil Modelleri

- RNN tabanlı yaklaşımlar
  - Tag ML: pre-trained (sonraki kelimeyi tahmin için eğitilmiş) modelin başka bir görevde ilk kullanımı
  - Elmo: çok sayıda görevde performans artışı
- Transformers tabanlı yaklaşımlar:
  - GPT: Sonraki kelimeyi tahmin
  - BERT: Maskelenmiş kelimeyi tahmin, sonraki cümle mi ikili sınıflandırma (pozitif : ardışık iki cümle, negatif: rasgele iki cümle)
  - RoBERTa: Maskelenmiş kelimeyi tahmin
  - AlBERT: Maskelenmiş kelimeyi tahmin, sonraki cümle mi ikili sınıflandırma (pozitif : ardışık iki cümle, negatif: pozitifteki 2 cümleyi yer değiştir) https://twitter.com/huggingface/status/1204808335922548737?s=03
  - ELECTRA: değiştirilmiş kelimeleri tahmin
  - BART: çeşitli görevler (sonraki slayt)

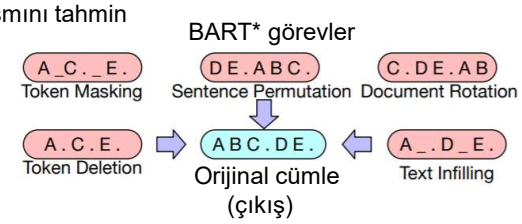
[*] https://www.youtube.com/watch?v=S-CspeZ8FHc&feature=youtu.be
[*] görsel karşılaştırma: https://twitter.com/Thom_Wolf/status/1186225108282757120?s=03





# Self - Supervised Görevler

- Büyük miktarda eğitim verisi üretmek için üretilen yan görevler
- İnsan etiketlemesine / emeğine ihtiyaç yok
- Sonraki kelime(ler)i / cümleyi / film karesini tahmin
- Gürültü ekleme
  - Maskelenmiş kelime(ler)i tahmin
  - Sırası bozulmuş cümleyi düzeltme
  - Harfleri bozulmuş kelimeyi düzeltme
  - Resmin silinmiş kısmını tahmin
  - vb.
  - Başka ??




[*] https://arxiv.org/abs/1910.13461
[**] https://lilianweng.github.io/lil-log/2019/11/10/self-supervised-learning.html

# Byte pair encoding

aaabdaaabac

ZabdZabac (Z=aa)

ZYdZYac (Y=ab Z=aa)

XdXac (X=ZY Y=ab Z=aa)

- Veri sıkıştırmak için önerilen bir yöntem
- Sinirsel dil modellerinde çok yaygın

[*] https://en.wikipedia.org/wiki/Byte_pair_encoding

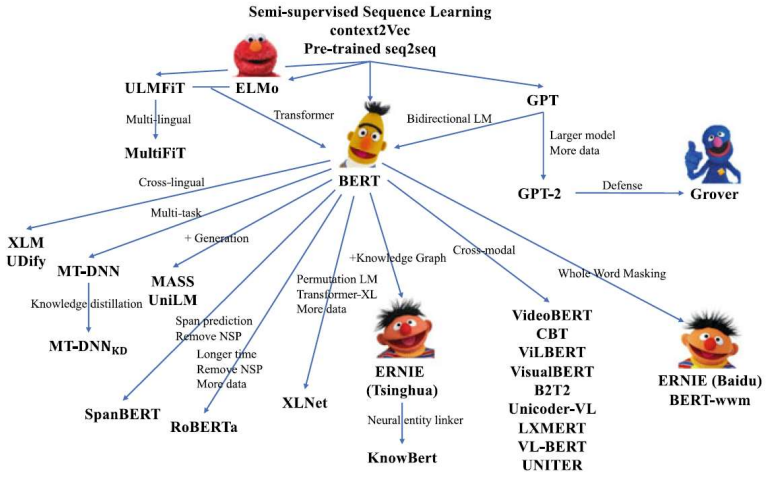




# Byte pair encoding

- Dil işleme için «Neural Machine Translation of Rare Words with Subword Units» https://arxiv.org/abs/1508.07909 ile popüler oldu.
- http://ufal.mff.cuni.cz/~helcl/courses/npfl116/ipython/byte_pair_encoding.html
- https://medium.com/@makcedward/how-subword-helps-on-your-nlp-model-83dd1b836f46
- Wordpiece, sentencepiece (free, fast)
- Türkçe için sadece OOV değil, normal kelimeler içinde çok önemli Neden?
  - gel-ecekmiş, bil-ecekmiş, bit-miş, gül-müştü

# Büyük Resim*




[*] Liu Z., Lin Y., Sun M. (2020) Sentence Representation.
In: Representation Learning for Natural Language Processing.
Springer, Singapore. https://doi.org/10.1007/978-981-15-5573-2_4





# Hazır Modeller

- https://github.com/pytorch/fairseq/tree/master/examples
- https://huggingface.co/models
- Dil modeli eğitmek: https://huggingface.co/blog/how-to-train
- Türkçe için:
- https://colab.research.google.com/drive/1qXX0Joz_tQMUFemJjx8QH9v-zWKNxJ-N#scrollTo=fbT9CyI7Ztj3 Ferhat Atlınar'a teşekkürler
- https://medium.com/@atlinar21/roberta-model-e%C4%9Fitimi-1cf59e8c4541
- BERTurk: https://huggingface.co/dbmdz/bert-base-turkish-cased
- Eksik kelime tahmini: https://colab.research.google.com/drive/1oFm-NTGNPNSU3EuA1dVOZTwj2HKqgzi7 Ahmet Bağcı'ya teşekkürler
- Metin sınıflandırma görevi: https://medium.com/@fsakyildiz/xlm-r-ile-metin-s%C4%B1n%C4%B1fland%C4%B1rma-fb6d99ab66ef Furkan Sami Akyıldız ve Burak Nuhbaşa'ya teşekkürler
- GPT2 Türkçe: https://huggingface.co/redrussianarmy/gpt2-turkish-cased

# Faydalı bağlantılar

- BERT ile kelime ve cümle temsilleri:
  - https://mccormickml.com/2019/05/14/BERT-word-embeddings-tutorial/
  - https://hanxiao.io/2018/06/24/4-Encoding-Blocks-You-Need-to-Know-Besides-LSTM-RNN-in-Tensorflow/#pooling-block
- BERT ile fine tuning: http://mccormickml.com/2019/07/22/BERT-fine-tuning/
- Attention and Augmented Recurrent Neural Networks https://distill.pub/2016/augmented-rnns/
- Transformers'sız dil modeli https://arxiv.org/abs/1911.11423
- Reformers: çoook geçmişe bakabilmek: https://ai.googleblog.com/2020/01/reformer-efficient-transformer.html?m=1
- Self supervised yönteme örnekler: https://amitness.com/2020/02/illustrated-self-supervised-learning/