Deep Sequence Models for Search Ranking - Session II

Tutorial Link: https://dlranking.github.io/dlrr/

Data source: https://huggingface.co/datasets/xglue

XGLUE: A New Benchmark Dataset for Cross-lingual Pre-training, Understanding and Generation)

Presenters: Moumita Bhattacharya,Abby Liu, Linsey Pang

Notebooks: Abby Liu, Linsey Pang

Date: August 14th, 2022

Agenda

- Brief History of Deep Learning Models For Search Ranking
- Sequence Models
  - Self Attention and Multi Head Self Attention
  - Transformer Architecture
- Deep Classifier
- Deep Siamese Network
- Hands-on Session

Key responsibilities of a search engine

Indexing — Ingesting and storing data efficiently so that it can be retrieved quickly

Querying — Providing retrieval functionality so that search can be performed by an end user

Ranking — Presenting and ranking the results according to certain metrics to best satisfy users' information needs

Brief History of Neural IR

- Neural IR resort to deep learning for tackling the feature engineering problem of learning to rank. □

- There have been some pioneer work, including DSSM [ref], and CDSSM [ref]. Followed by, DRMM [ref] and DeepRank [ref].

- Studies such as DRMM [ref] and DeepRank [ref] argued that DSSM and CDSSM only consider the **semantic matching** between query and document (similar to pure NLP tasks), but ignored **relevance matching**

Learning Deep Structured Semantic Models for Web Search using Clickthrough Data (DSMM: 2013)

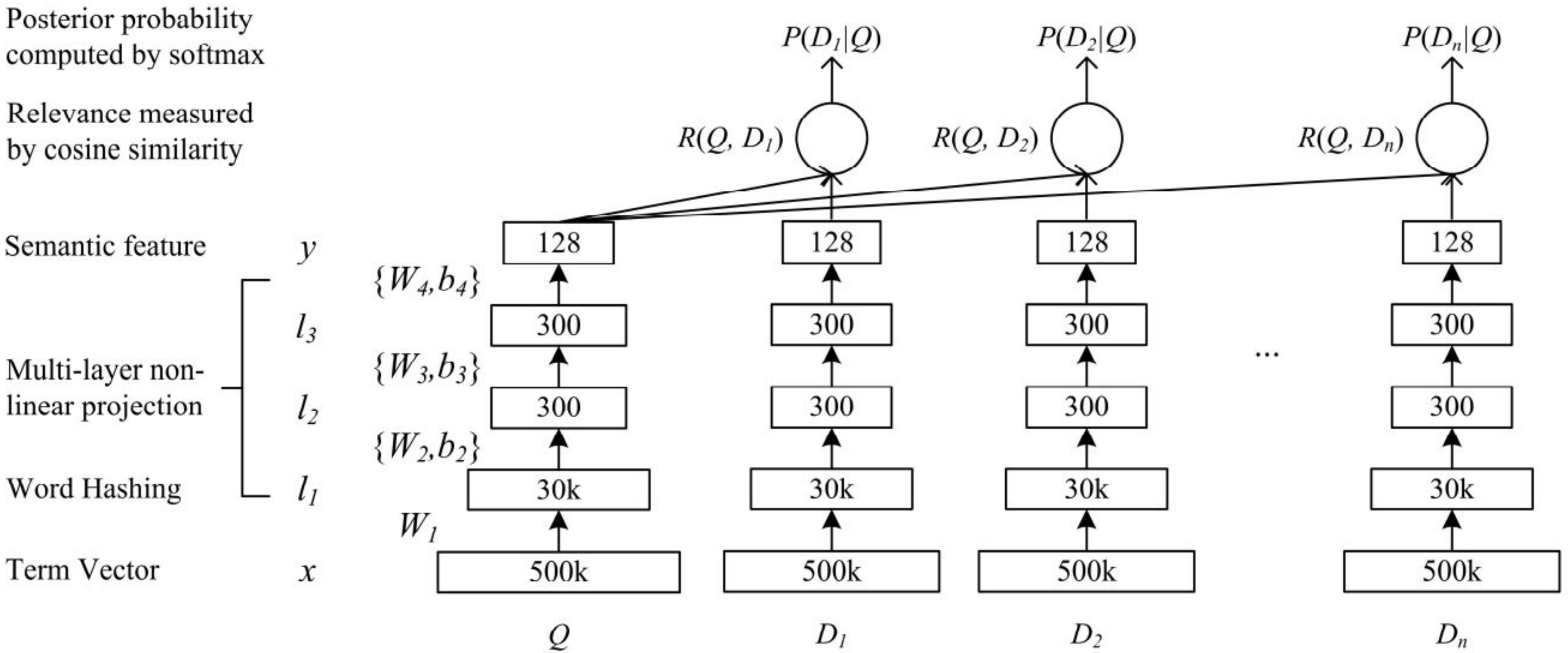


Figure 1: Illustration of the DSSM. It uses a DNN to map high-dimensional sparse text features into low-dimensional dense features in a semantic space. The first hidden layer, with 30k units, accomplishes word hashing. The word-hashed features are then projected through multiple layers of non-linear projections. The final layer's neural activities in this DNN form the feature in the semantic space.

A Deep Relevance Matching Model for Ad-hoc Retrieval (DRMM: 2017)

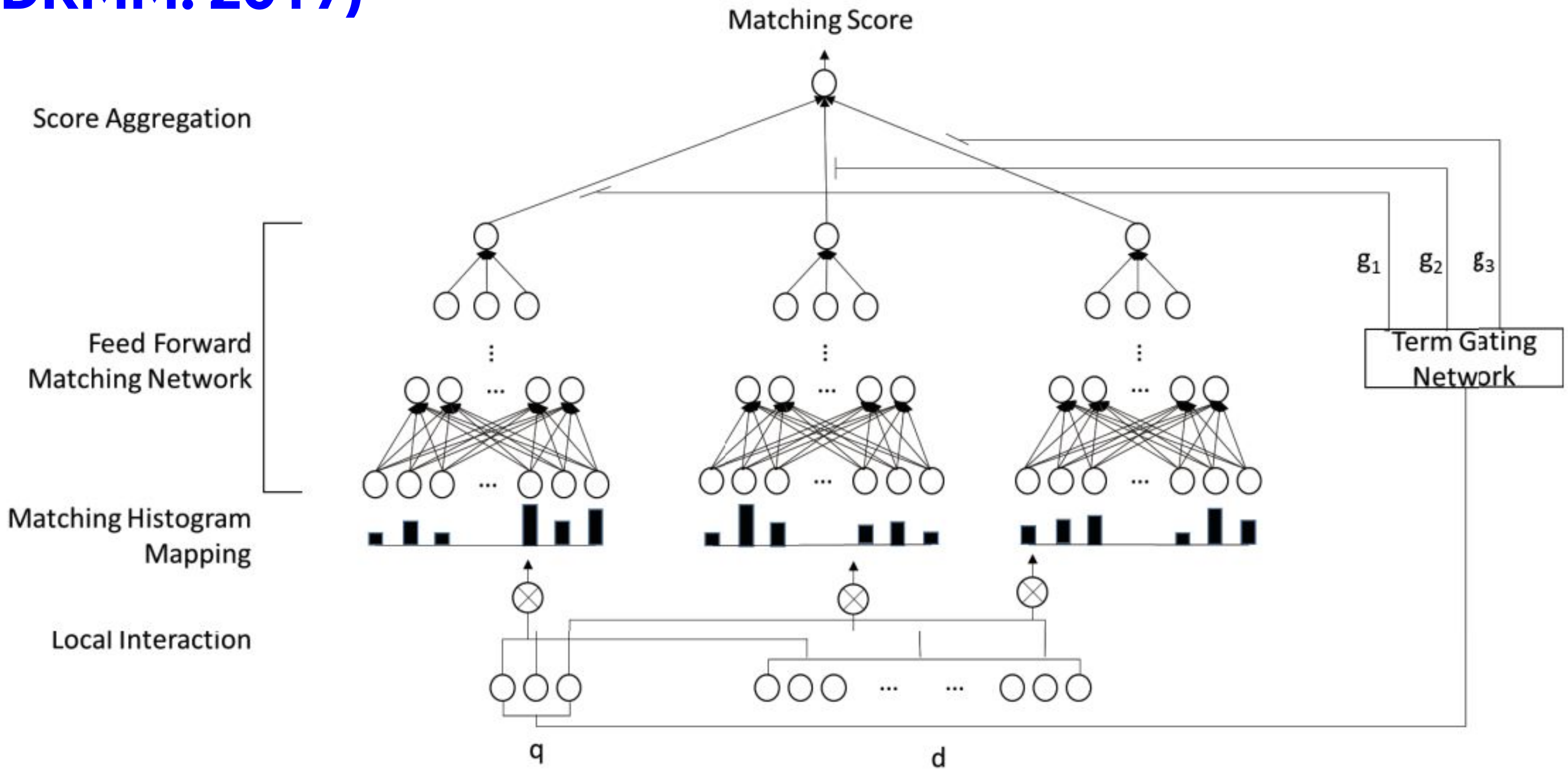


Figure 2: Architecture of the Deep Relevance Matching Model.

DeepRank: A New Deep Architecture for Relevance Ranking in Information Retrieval (2019)

Ref: (link)

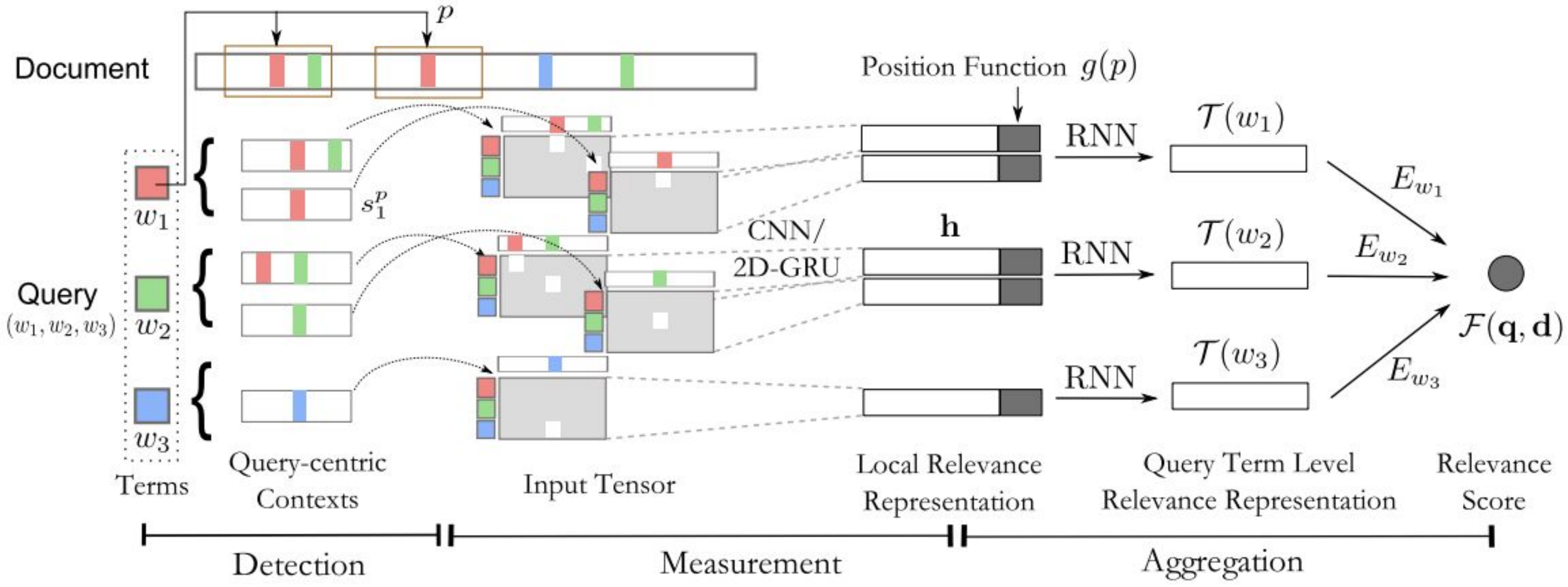
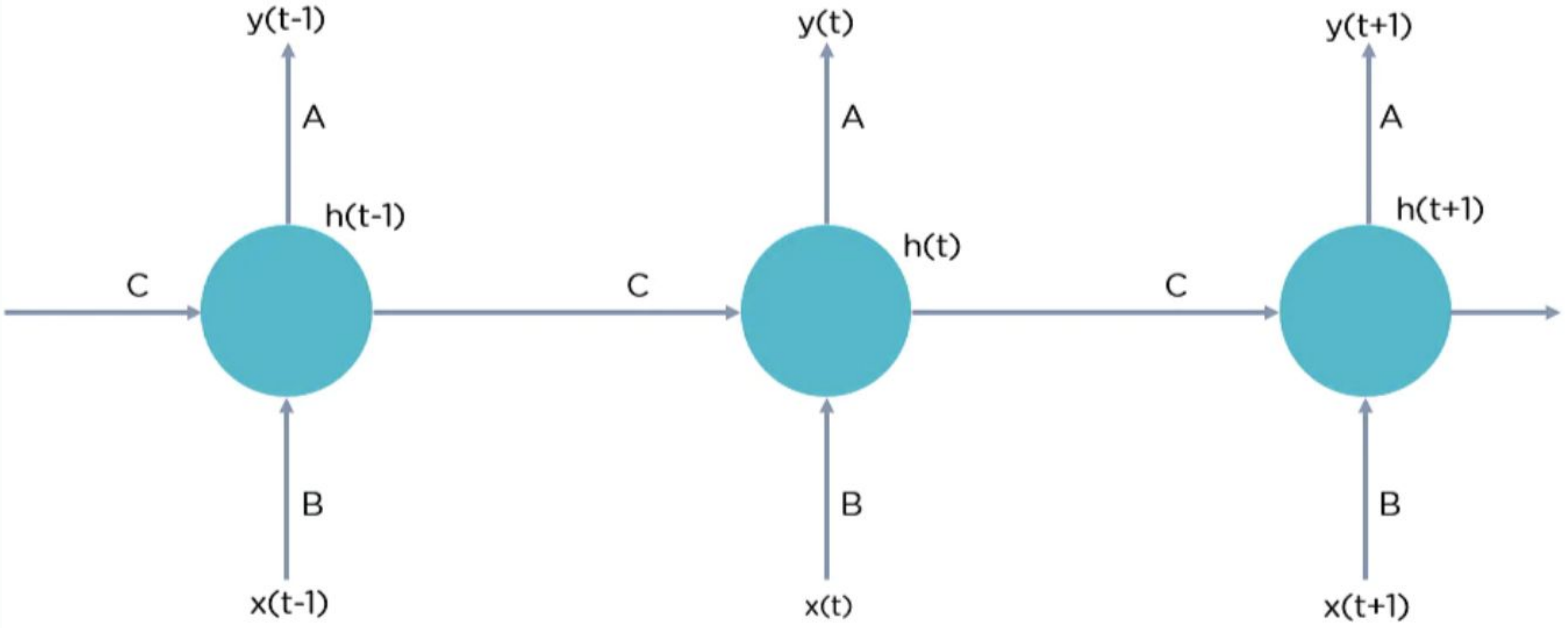


Figure 1: An illustration of DeepRank.

Sequence Models

- RNN, GRU
- Attention and Multi-attention
- Transformer

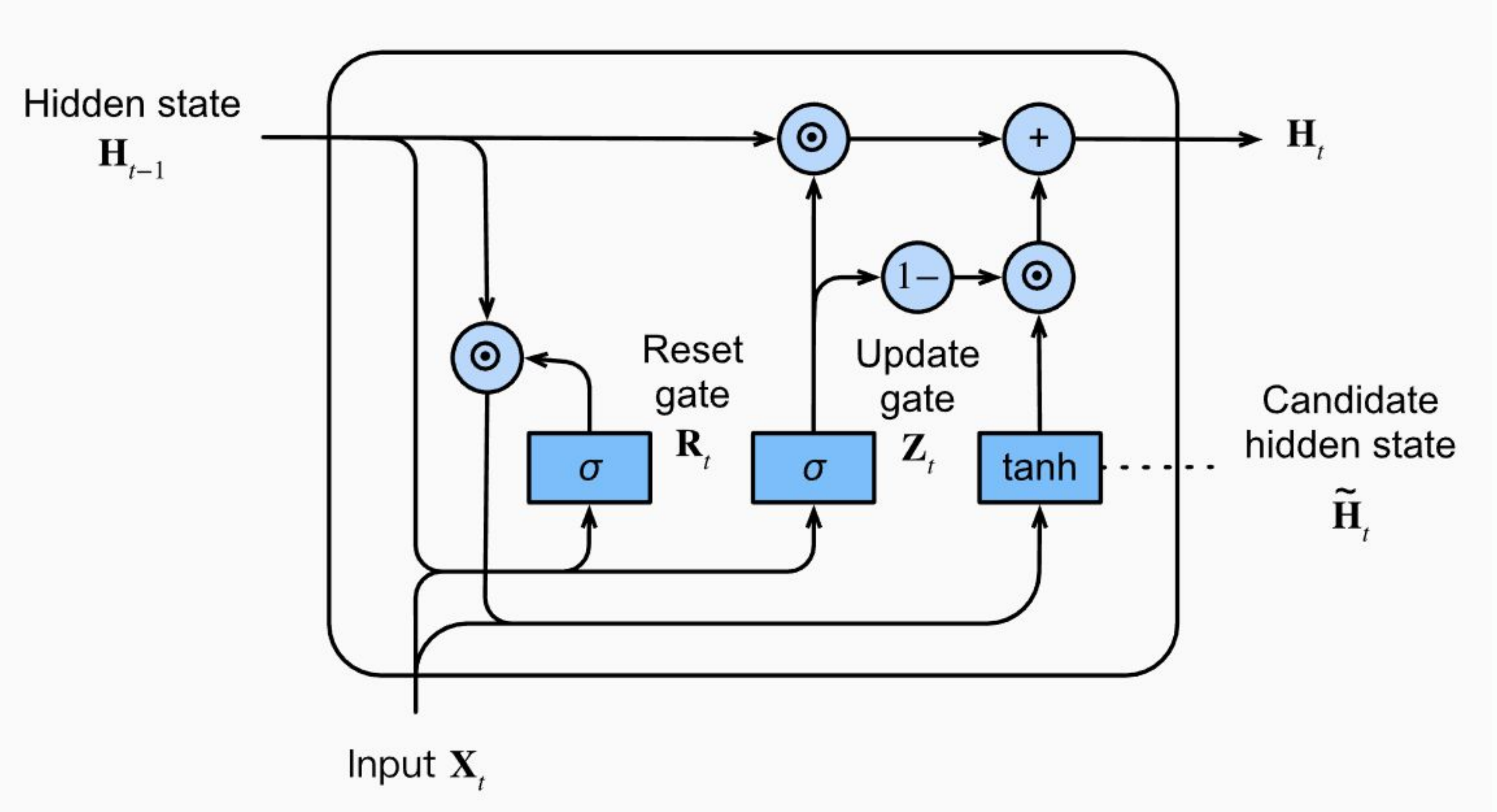
Recurrent Neural Network



$$h(t) = f_c\,(h(t\text{-}1),\, x(t))$$

$h(t)$ = new state
f_c = function with parameter c
$h(t\text{-}1)$ = old state
$x(t)$ = input vector at time step t

GRU


Hidden state
$\mathbf{H}_{t-1}$
$\mathbf{H}_t$
1−
+
Reset
gate
$\mathbf{R}_t$
Update
gate
$\mathbf{Z}_t$
σ
σ
tanh
Candidate
hidden state
$\tilde{\mathbf{H}}_t$
Input $\mathbf{X}_t$

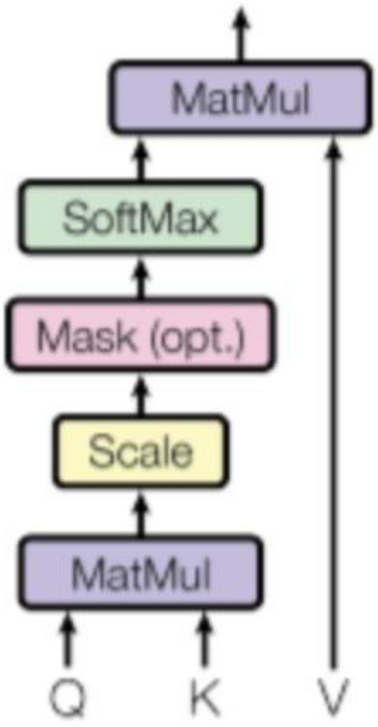
$$\tilde{c}^{<t>} = \tanh(W_c[\Gamma_r * c^{<t-1>}, x^{<t>}] + b_c)$$

$$\Gamma_u = \sigma(W_u[c^{<t-1>}, x^{<t>}] + b_u)$$

$$\Gamma_r = \sigma(W_r[c^{<t-1>}, x^{<t>}] + b_c)$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) + c^{<t-1>}$$

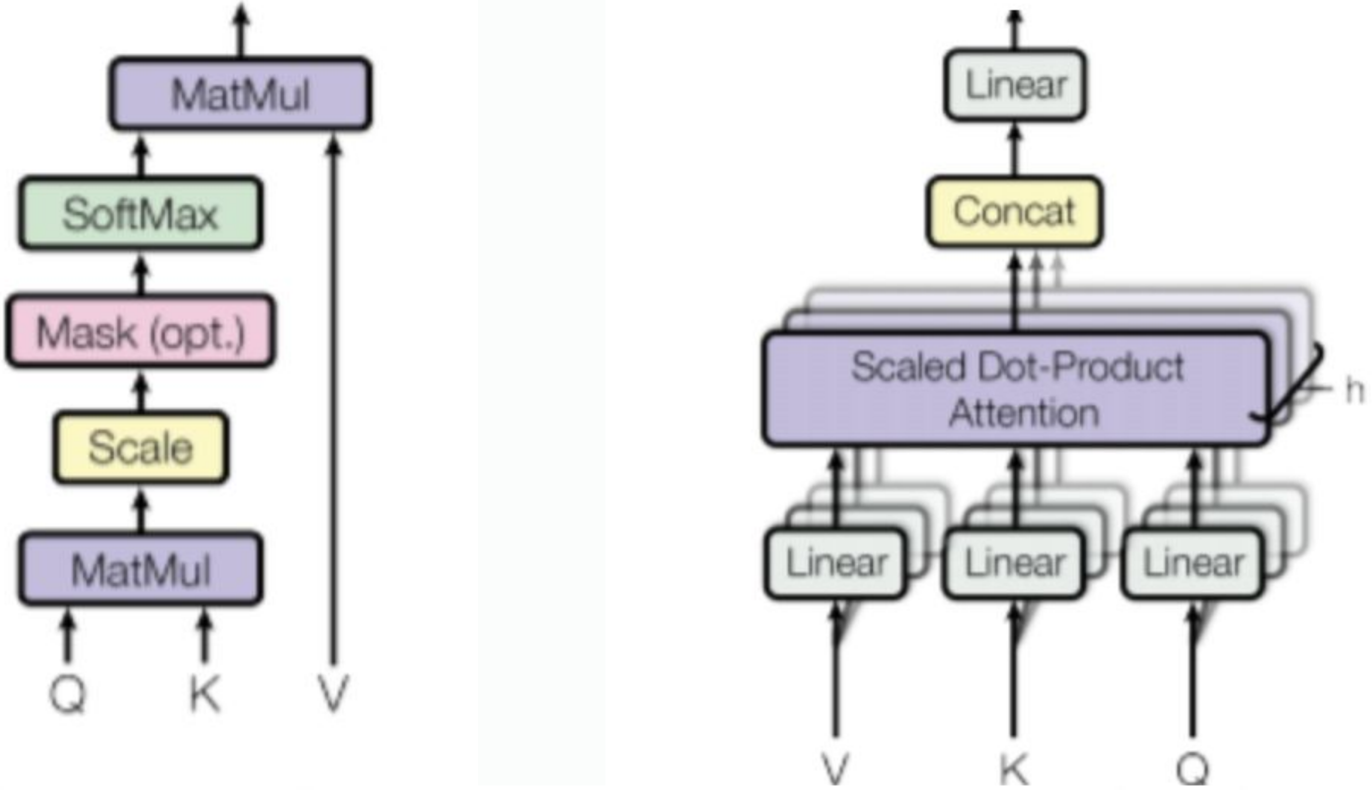
Self and Multi-Head Attention



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

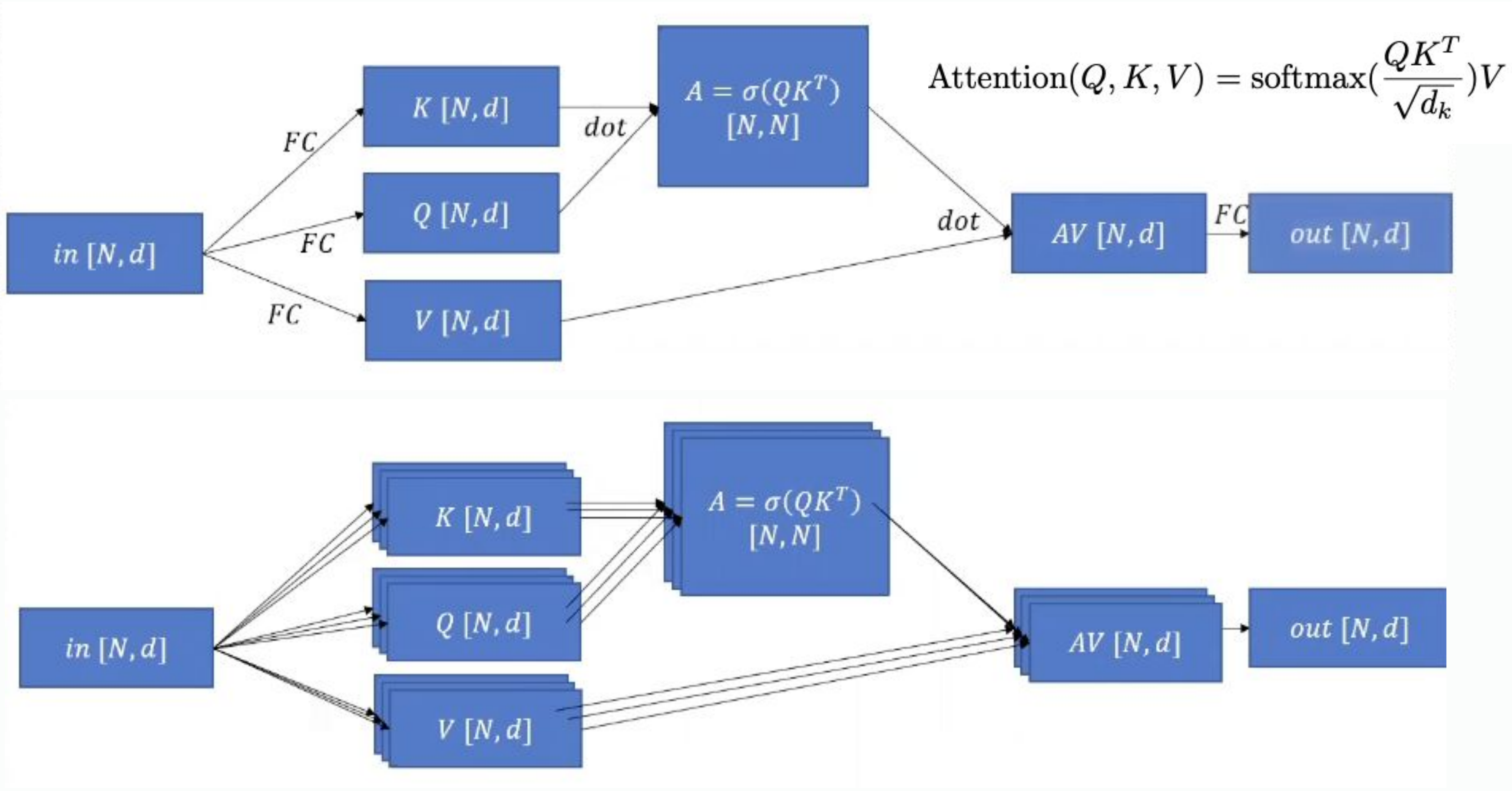
- For each word a attention vector is created.

Self and Multi-Head Attention



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

Single and Multi-Head Attention



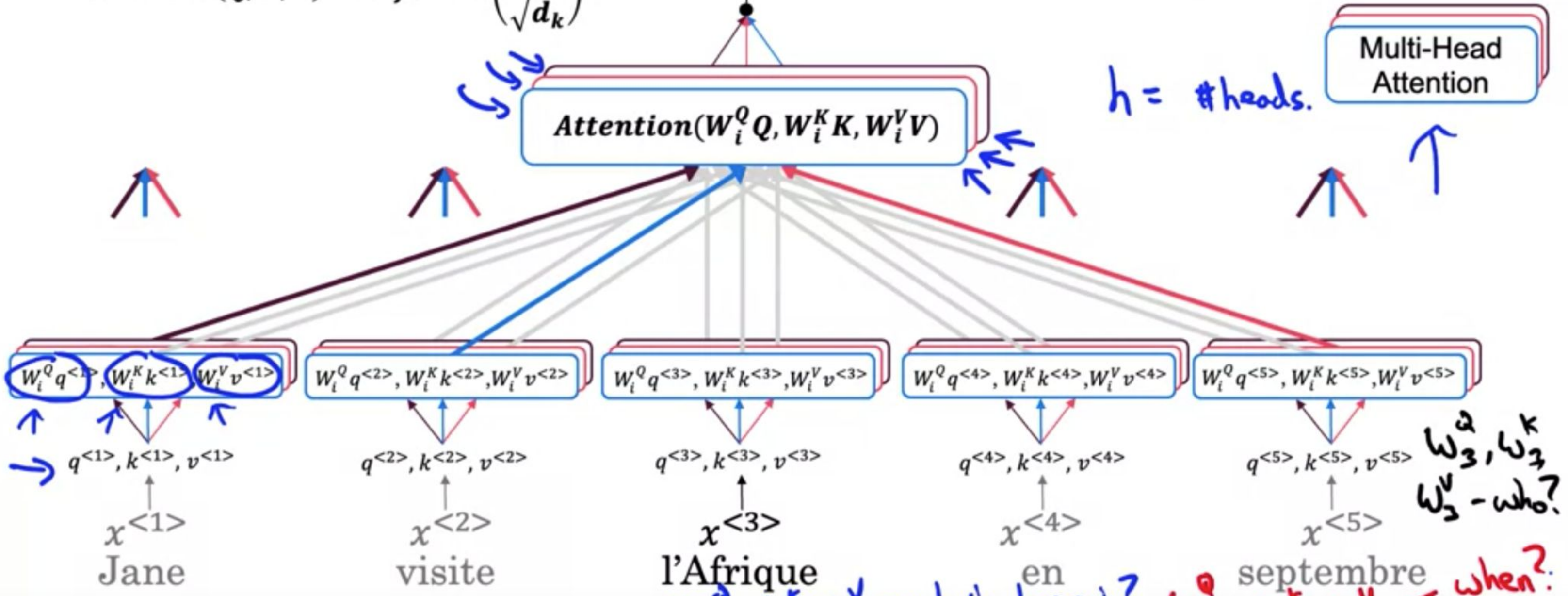
Multi-Head Attention

"head"

$$MultiHead(Q,K,V) = concat(head_1 head_2 \dots head_h)W_o$$

$$head_i = Attention(W_i^Q Q, W_i^K K, W_i^V V)$$

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



W_1^Q, W_1^K, W_1^V - what's happening? W_2^Q, W_2^K, W_2^V — when?

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

Andrew Ng

Transformer - Model Architecture

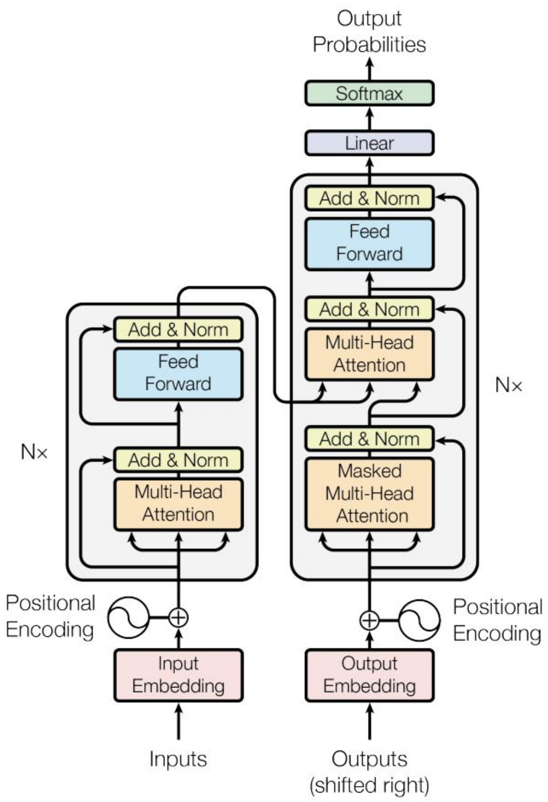


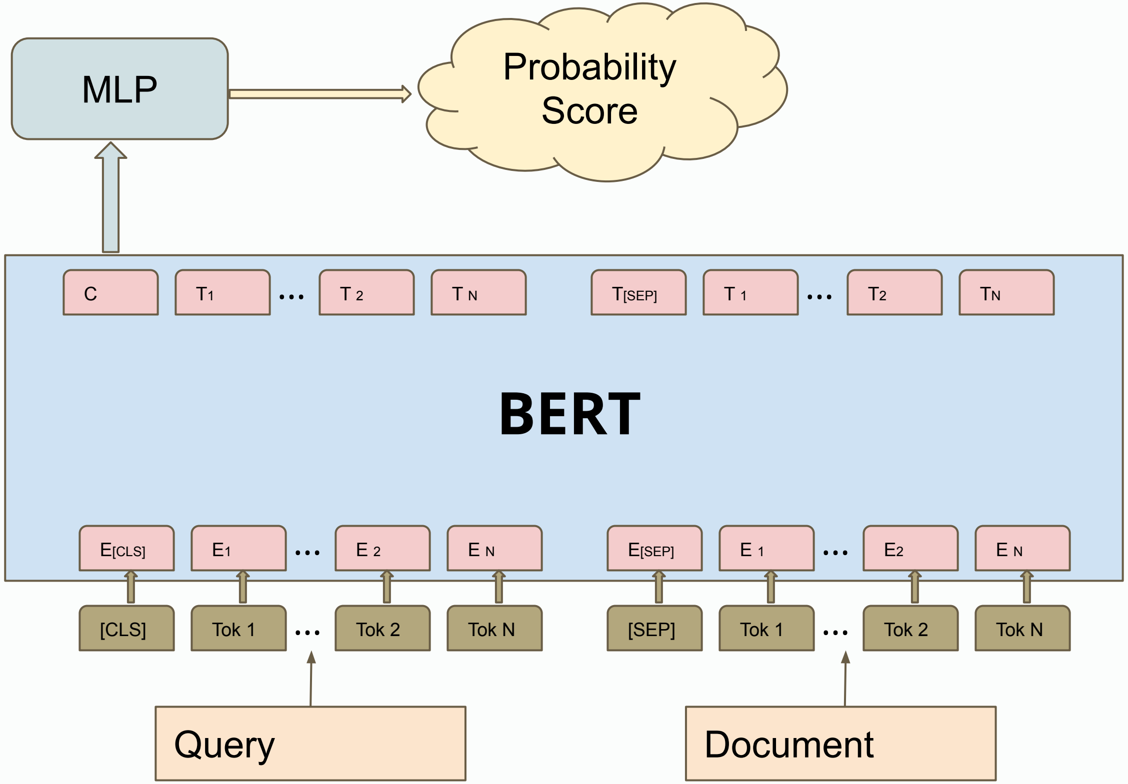
Figure 1: The Transformer - model architecture.

1. **Encoder**:
   a. Input Sequence Embedding
   b. Positional Encoding
   c. Multi-head attention
   d. Feed forward
2. **Decoder**
   a. Output Sequence Embedding
   b. Positional Encoding
   c. Multi-head Attention
   d. Encoder-Decoder Attention Block
   e. Feed forward
3. Linear: Feed Forward Network
4. Softmax: transforms into a probability distribution

Deep Classifier Network

- The goal is to rank the documents based on the query document similarity
- Use the transformer based BERT model to generate the embeddings of the queries and embeddings of the documents that capture the semantic meanings of them from the text
- Concatenate the query and document embeddings, and feed into a classification network to calculate the probability of the document related to the query
- Rank the documents based on their probability score to the query

Deep Classifier Network



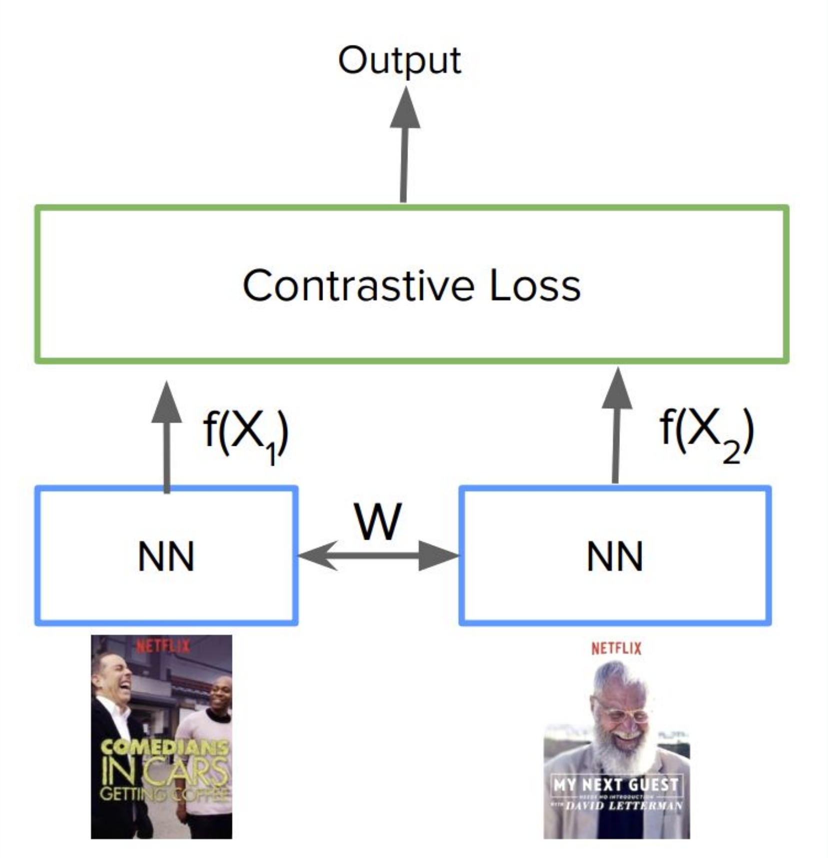
Deep Classifier Network

[Colab Demo]

Siamese Network and Metric Learning

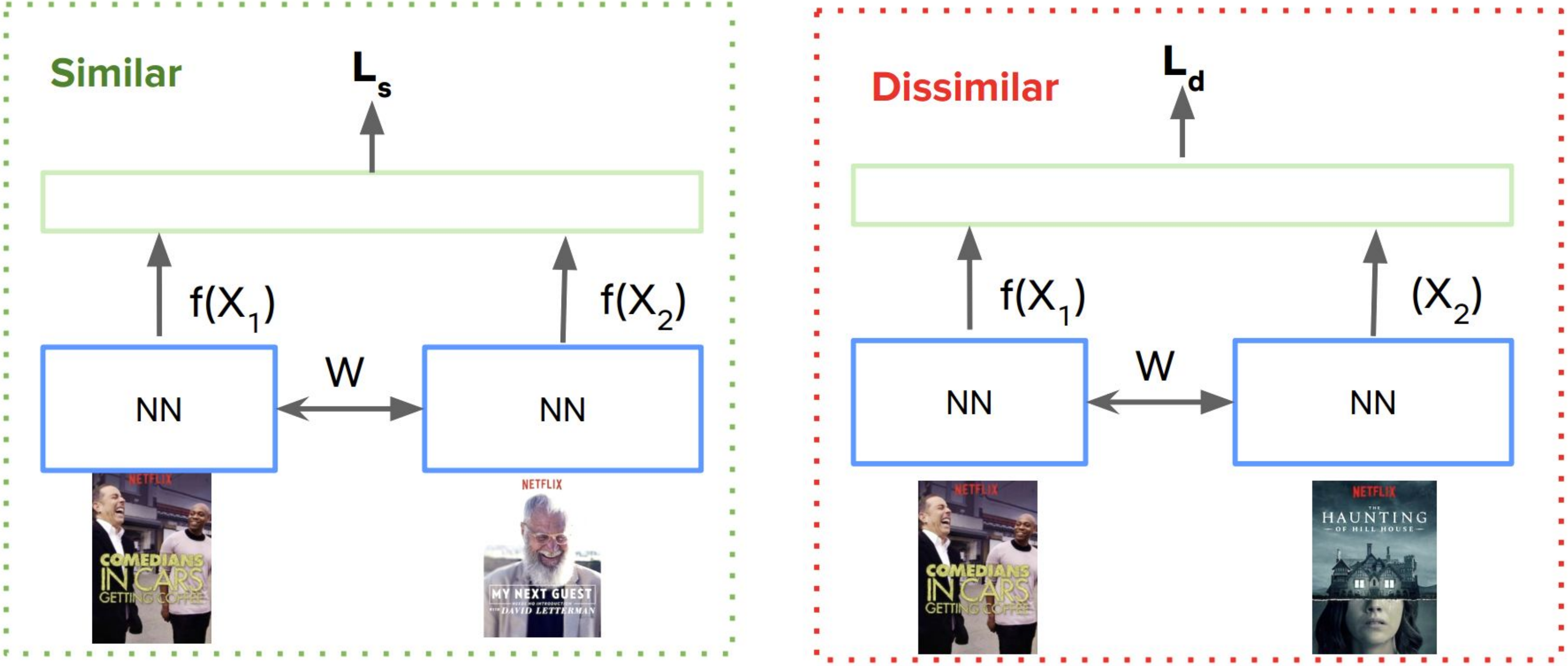
- Siamese network are networks that compare similarity of two inputs
- The goal is to capture similarity between embeddings, such that the projected distance of similar items in the embedding space is smaller than the dissimilar items
- Compared to the standard distance metric learning, it uses deep neural networks to learn a nonlinear mapping to the embedding space
- Helps with extreme classification settings with huge number classes, not many examples per class

Siamese Networks



- Left and right legs of the network have identical structures (siamese)
- Weights are shared between the siamese networks during training
- Networks are optimized with a loss function, such as contrastive loss

Siamese Networks



$$L = \sum_s L_s + \sum_d L_d$$

Total loss is the sum of losses on similar pairs and dissimilar pairs

Contrastive Loss

The goal is to minimize L with respect to W, such that D_W is small for similar pairs, and large for dissimilar pairs

$$L^i = (1-Y)L_s(D_W^i) + YL_d(D_W^i)$$

Y is set to 0 if pairs are similar otherwise 1

Exact loss is defined as:

$$L = (1-Y)\frac{1}{2}(D_W)^2 + Y\frac{1}{2}\{\max(0, m - D_W)\}^2$$

m > 0 is the margin

Triplet Loss

The goal is the same, the distance between similar items must be low, and dissimilar items must be high.

$$L = \max(D(a,p) - D(a,n) + \alpha, 0)$$

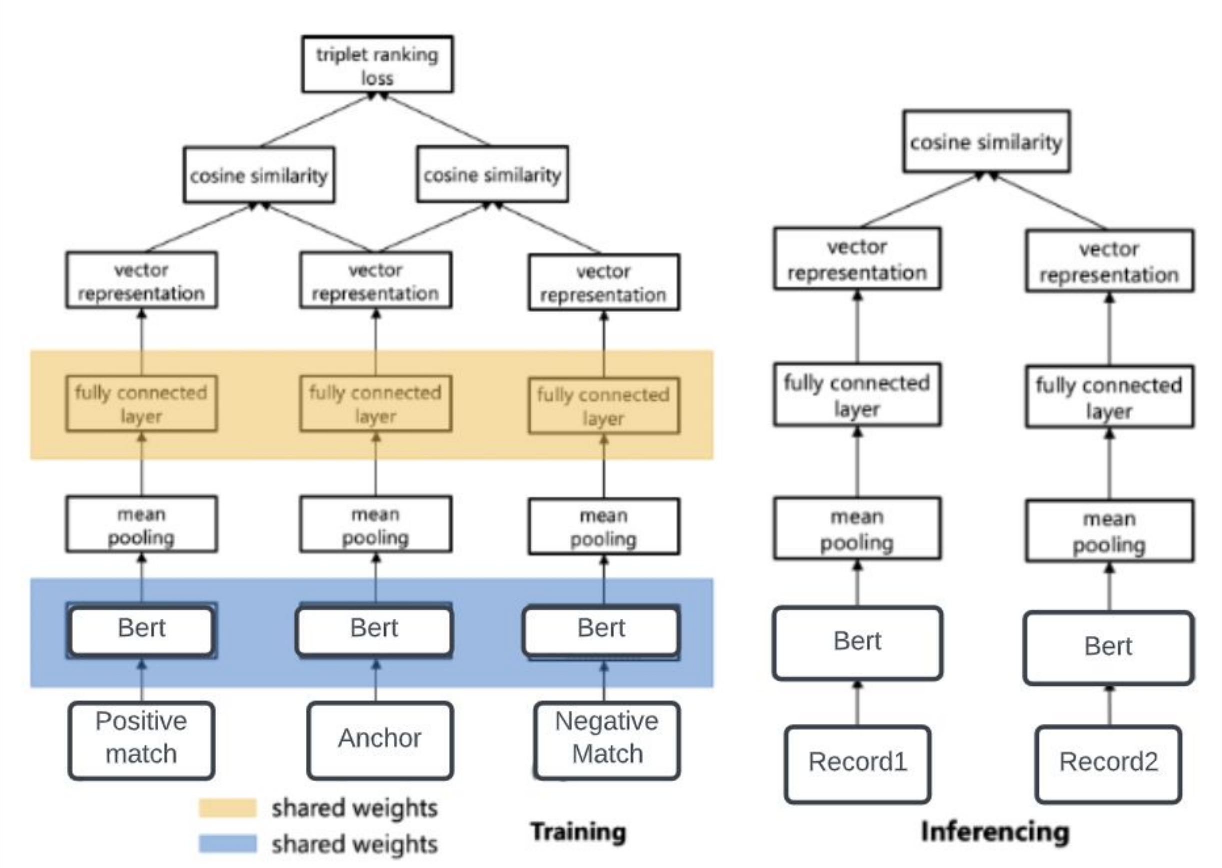
Loss of a triplet (a, p, n)

When using an Euclidean distance:

$$L = \frac{1}{Z}\sum_{i}^{N}\left[||f(x_i^a) - f(x_i^p)||_2^2 - ||f(x_i^a) - f(x_i^n)||_2^2 + \alpha\right]_+$$

$[.]_+ = \max(0, .)$
Hinge loss function

Deep Siamese Network



Triplet Loss Architecture

Deep Siamese Network

Similarity function:

$$s(x, y) = \frac{e(x) \cdot e(y)}{\|e(x)\| \|e(y)\|}$$

where $e()$ is the function that projects the raw input to an embedding vector (the sub-network in the Siamese network) and $\|e(x)\|$ denotes the norm of the vector $e(x)$.

Cost function:

$$L_{triplet} = \frac{1}{|X|} \sum_{(a,p,n) \in X} max(|s(a, p) - s(a, n) + \alpha|, 0)$$

where X is the set of (a, n, p) triplets and α is a margin between positive and negative pairs. The triplet loss pushes $s(a, p)$ towards 1 and pushes $s(a, n)$ below than $s(a, p)$ by at least α. We set $\alpha = 0.3$ for the output shown in this work.

Deep Siamese Network

Demo

Let’s work on the hands on examples now!!