Zhang - CH11 - Attention & tranform

① Queries & keys & Values
② Attention Pooling by similarity
③ Attention scoring
④ Bandanau Attention mecnanism
⑤ multi-head attention
⑥ Self-Attention & Positional encoding
⑦ transformer architecture
⑧ tranformers for vision
⑨ large Scale pre-train w/ transform

---

early years of Deep learning boom were in MLP, convolutional & recurrent

there were plenty of methodological innovations but the architectures stayed mostly the same

Relus, residual layers, batch norm, dropout, adaptive learning

(CV) (NLP)
CNN & LSTM style architectures remained State-of-art

today transformers dominates NLP

NLP
given a new task, the ~~dear~~ default approach is to grab a pre-trained transformer

BERT, Electra, RoBERTa, longformer

Adapt outlayers to task
& fine-tune on avaible data

transformers are based on the attention mechanism

original an enhancement to encoder-decoder RNNs

Recall that first Seq-2-Seq models the input was compressed into a single fix-length vector

the intuition behind Attention:

- instead of compressing
- Might be better for the decoder to revisit the input seq at every step

Also refer to diff points of the input @ DIFF points of the decode

Attention = means to dynamically attend to different parts of the input @ each Decode step

high-level = encoder would produce a rep representation vector of equal length to the original input seq

decoder receives the input as a context vector — A weighted sum of each of the representations of the input @ each step

- weights determine to extent to which a given portion of the input is considered

  each step "focuses" on each token

  key goal is for the weight assigning proc to be differentiable so it can be learned along w/ the weights & biases of A neural network

- Attention was very succes @ outperforming the Dom RNN architechrs

  ↳ Also Attention weights provides insight of salient features

Vaswani (2017) proposed transformer arch dispensing w/ recurrent connections

↳ instead relying on clevery arranged Attention mechanisms to capture relationships between input & output tokes

Performed well & by 2018 = SOA NLP

Moreover, same time the main practice became to pretrain large-scale models on enormous generic background corpa to optimize some self-supervised pretrain objective

↳ the to fine-tune using availble data down-stream

With this pretrain method, the gap between transformers & ~~RNNs~~ original architectures grew

large Pretrained models are known as foundation models

11.1 Queries, keys & values

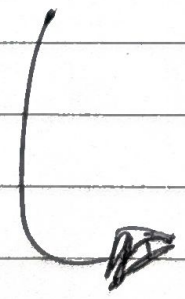
Prev archs such as CNNs & RNNs start inputs of defined size (224 x 224 e.g.)

Variable size has be dealt w/ by processing one token @ a time

this can lead to significant problems when the input is of vary size & with vary input quality

In particular, for ~~some~~ long segs it can be difficult for the model to keep track of everything ~~that~~ has be viewed & generated by the network

Attention Mechanism



Attention Mechanism

① = $\{(k_1, v_1, \ldots, k_m, v_m)\}$

Define attention as:

$$Att(q, D) \sum a(q, k_i) v_i$$

q = query

$a(q, k_i)$ = Scalar attention weights

often referred to as attention pooling

if an a is high then the model pays particular attention to the terms

↳ this is where the name "attention" comes from

Attention over D gens a lin comb of values contained in the database

if $a(q, k_i) = 1$ then this is the same as a direct trad database ϕ

normalize weights to ensure they add up to 1

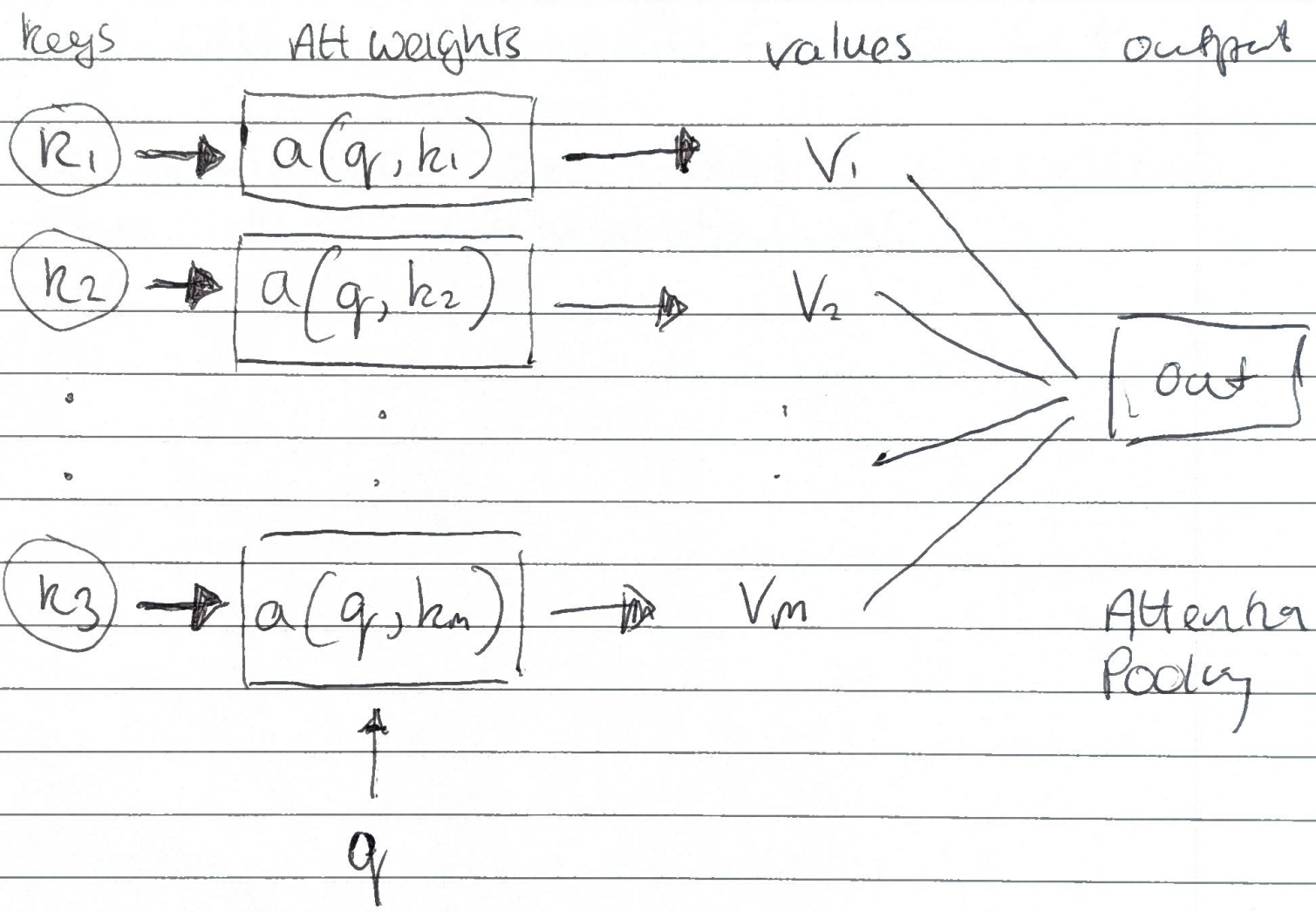
$$a(q, k_i) = \frac{a(q, k_i)}{\sum a(q, k_j)}$$

↳ Can also exp to ensure not neg

this operation is Differable & never vanish

↳ Desireable properties of a model

Can create a non-Diff Model that can be trained via re-enforcement learning



Att Mech computes a linear combi over values V_i

weights are Derived accord to compat between q & k_i

Attention mechanism allows us to aggregate data from many (key, val) pairs

Att mech provides a differentiable means of control by which a NN can select elements from a set

to construct an assoc weighted sum over representations

11.2 Attention pooling by Similar

Example applies Att in a reg & classif setting via kernel density estimation

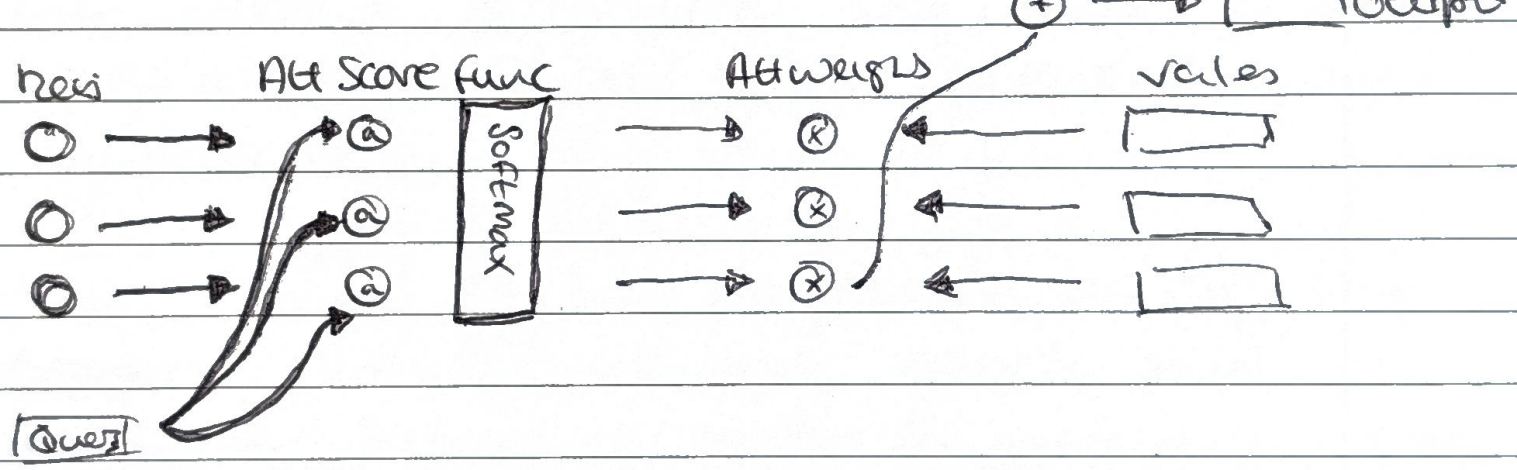
Skipped

11.3 Attention Score Functions

(a)

Distance functions are more expensive than dot prods

So much work has gone into Att scoring



Dot Product Attention

Skipped

11.4 Bahdanau Attention Mechanism

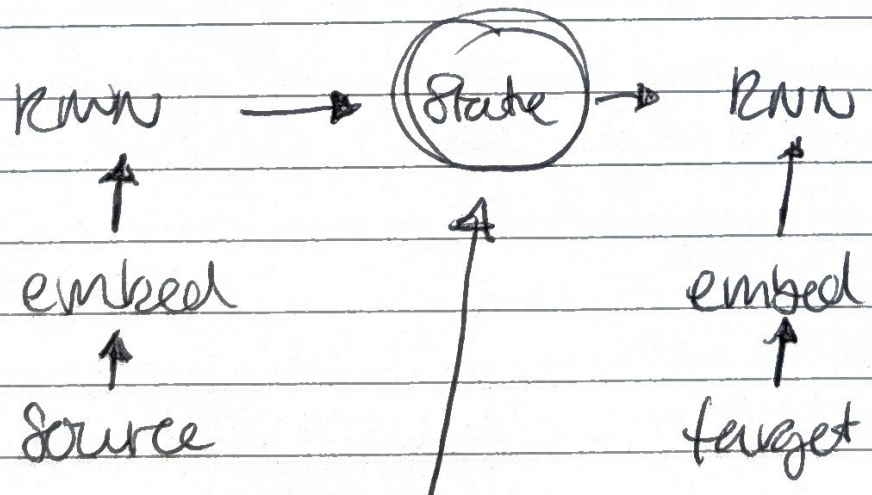
In the previous chapter we looked @ encoder-decoder arch for seq2seq based on RNNs (Sutskever, 2014)

↳ encoder transforms variable length seq into fixed shape context var

↳ decoder generates the output seq token-by-token based on the generated tokens & context variable

Conventionally, in RNNs all relevant info about a source sequence is translated into a fixed dimension vector by encoder

this fixed state is pick up by the ~~encoder~~ Decoder as the sole and exclusive source of info for generating the sequence



the encode-Decode is reasonable for short seq

but infeasible for long

Soon run out of "space" in the interned representation to store all important info

Consequently Decoder will fail to translate long & complex seqs

Dzmitry Bahdanau et al (2014)

↳ when pred, if not all input tokens are relevant

the model attends to only part of the input sequence

that are deemed relevant for prediction

used to update current state

before next generated token

key point =

C_t = context vector state — each

Dynamically update this @ token gen

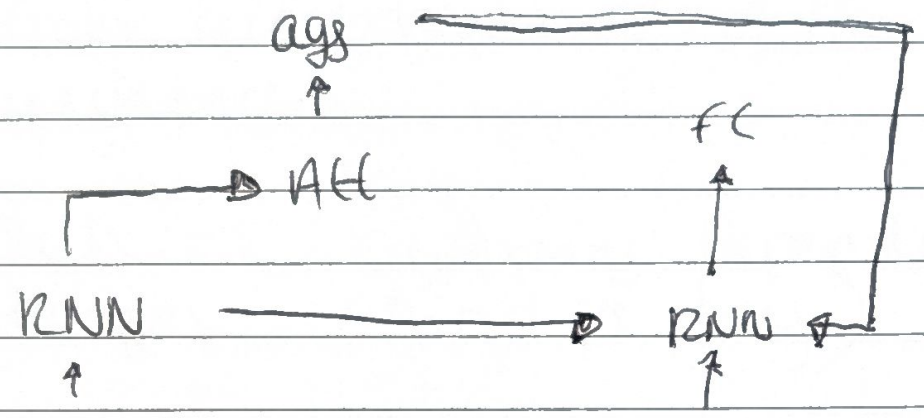
↳ using (1) original text h_t

(2) Decoder generated S_{t-1}

yields C_t which is what Decoder uses

Context variable is the output of the attention pooling $c_{t'}$

$$c_{t'} = \sum a(s_{t'-1}, h_t) h_t$$



this model was later updated to include Att at the generate token too

Summary

When pred a token

↳ if not all input tokens are relevant

↳ RNN ~~for~~ encoder-Decoder w/ Att Mech

Selectively aggregates diff parts of the input seq

Achieved by treating the "state" as additive attention pooling

Decoder hidden state at previous time step = Query

encoder hidden state at all time steps as both keys & values

11.5 Multi-head Attention

given same set of queries, keys & values

we may want model to combine knowledge from Diff behaviours of the same atten mechanism

such as capturing Dependancies of various ranges (short vs long)

Allow Att Mech to use Diff representation subspaces of Queries, keys & values

Instead of performing a single Attention Pool

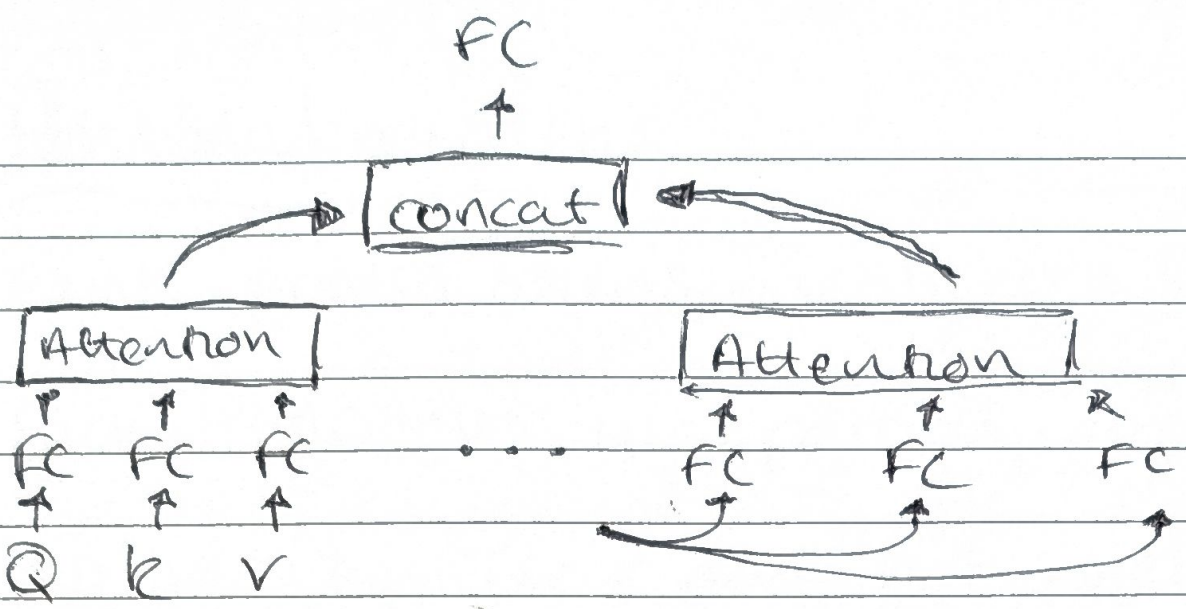
Queries, keys and values can be transf w/ h indep learned linear projections

these are then fed to the Attention pooling in parrell

the Att pooling outputs are concatened & transformed

↳ w/ another linear proj to form output

Multi Attention pool, where each Att pool output = A head



~~Multi-Head Attention~~

11.6 Self-Att & Positional encoding

Imagine Feeding Sequence of tokens into an Att Mech such that @ every step each token has its own Query, keys & values

When computing the value of a tokens representation @ the next layer

the token can attend (via its Query) to any other tokens

Using the full set of Query-key comp score

for each token, we can compute A rep by building the approp weighted sum over all the tokens.

this is DIFF to Prev Sect
before Decoder Attend to encoder step
here each token attend to every other token

Positional encoding

RNNs process tokens one-by-one

Self-Attention uses parrel computation

Self-Att by itself does not preserve order of the sequence

What to do if we need order?

Represent this to the model as additional input to each token

these are called positional encodings

these can either be learned or fixed

Pos encodings can be based on sine/cosine functions for example

11.7 transformer Architecture

earlier Self-Att models rely on RNNs for input representations

the transformer model is solely based on attention mechanisms w/out any CNN or RNN layer

Originally proposed for Seq-to-Seq on text data

later applied to wide areas lang, visn, speech RF

transformer Lite has an encoder-decoder structure

but there is no NN

just modules based on Self-Att & positional encodings

Skipped as rest has no connection to NNs

11.8 Vision transformer

Skipped

11.9 large Scale Pretrain w/ transformers

All examples so far has includes coding models from scratch to perform specific task

each model becomes a specific expert that is sensitive to even slight shifts in the data dists

for better generalized models Pretrain on large data has become increasingly common