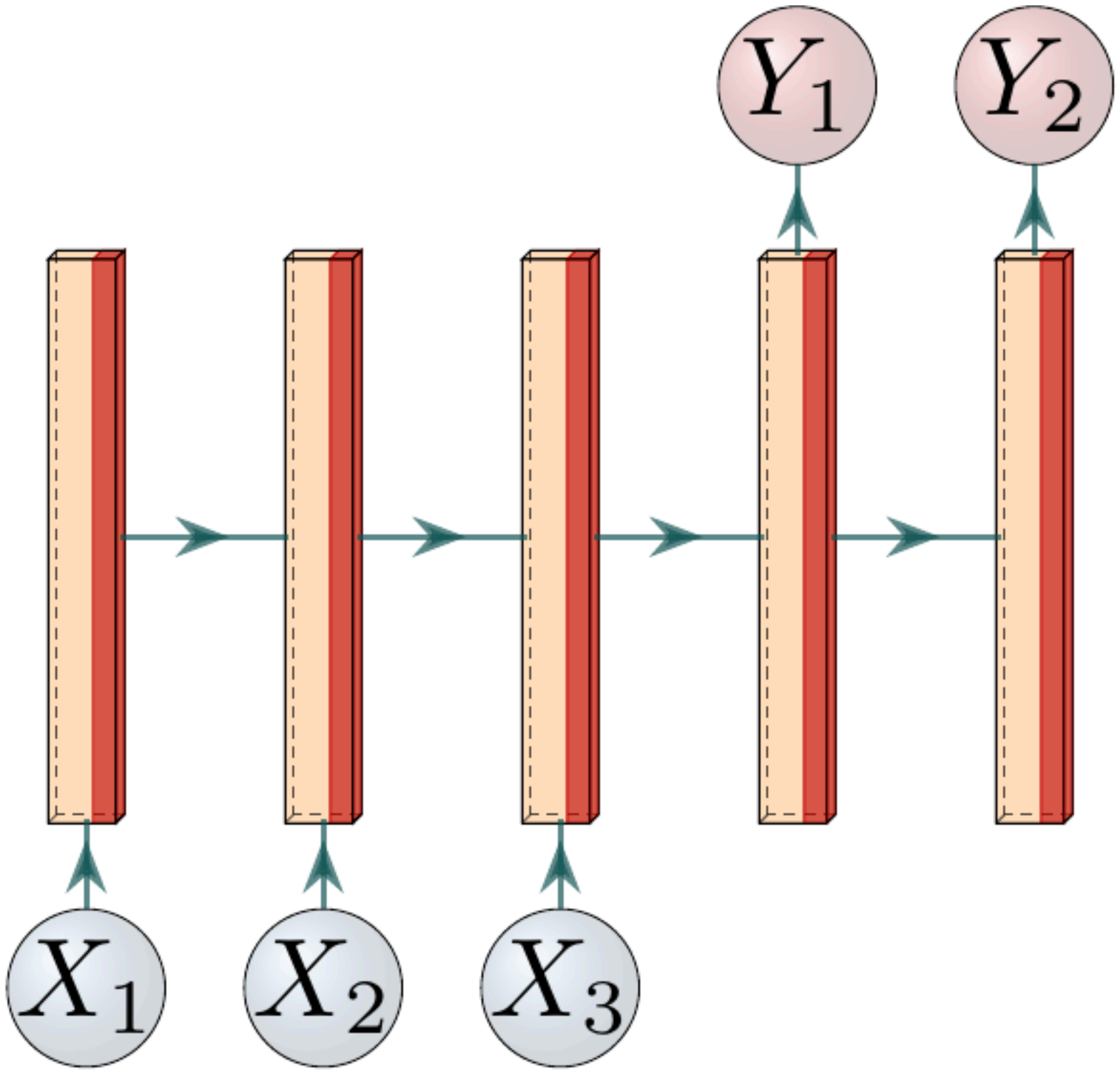
17-10–2024

Admin stuff

- Next project update on 26-10-2024
- Class test: 05-11-2024
- Any other?

Problem in general RNN

- Can you see any problem ?
- Longer sequence
  - Unable to capture long dependencies
  - Vanishing gradient problem

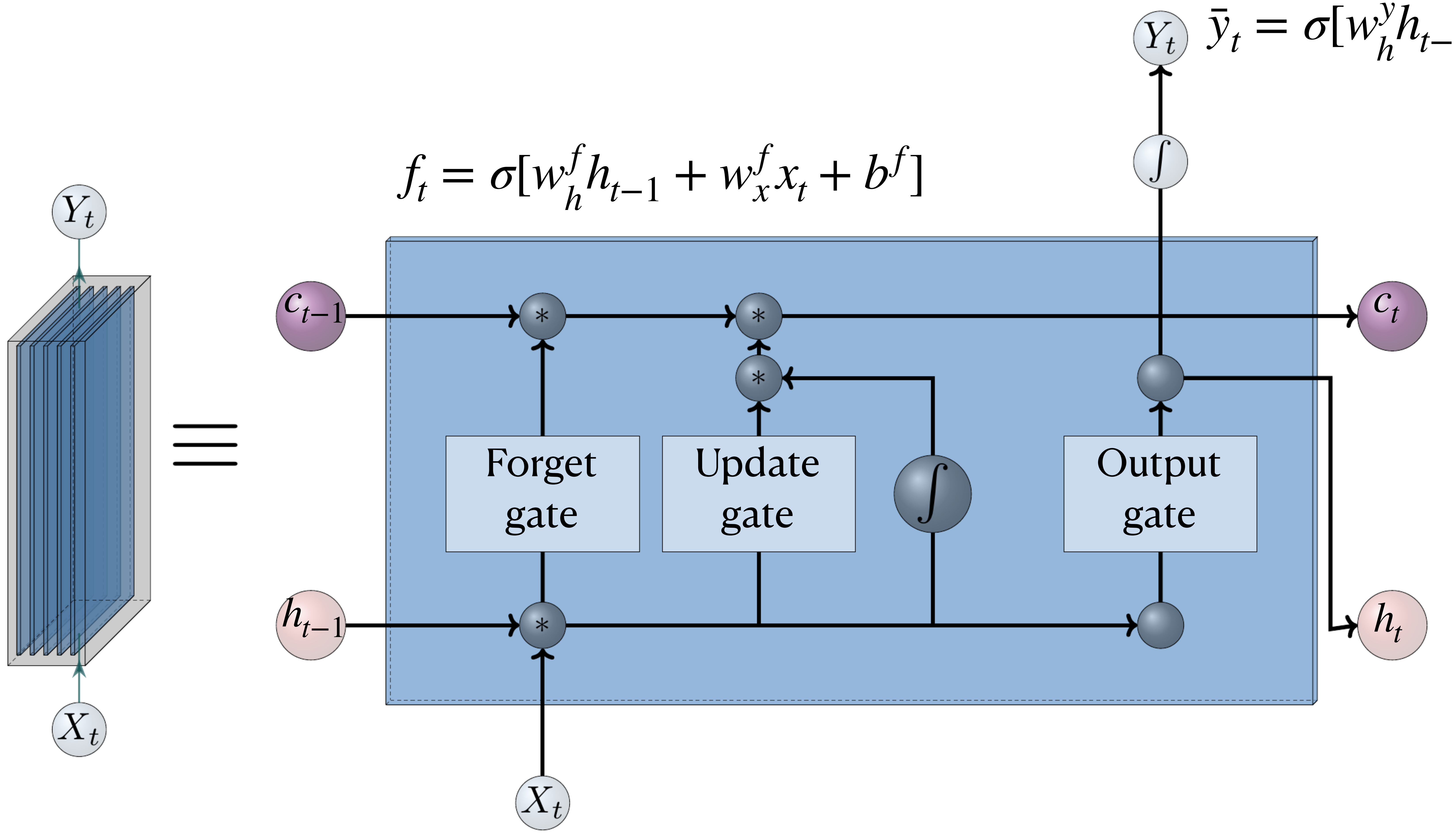


Observation in RNN

- RNN tried to capture the dependency at time t using all the past informations
  - Not all the past informations are important
    - Some are important (to remember)
    - Some are not (to forget)
- Can we build such a network/model which tells us at time t
  - Which information are important (to remember) ?
  - Which are not (to forget) ?
  - Also overcome the vanishing gradient problem?

Long short-term memory (LSTM)

- Cleaver way to overcome vanishing gradients in RNN



$$\bar{y}_t = \sigma[w_h^{\bar{y}} h_{t-1} + w_x^{\bar{y}} x_t + b^{\bar{y}}]$$

$$f_t = \sigma[w_h^f h_{t-1} + w_x^f x_t + b^f]$$

$$i_t = \sigma[w_h^i h_{t-1} + w_x^i x_t + b^i]$$

$$\bar{c}_t = \tanh[w_h^c h_{t-1} + w_x^c x_t + b^c]$$

$$c_t = f_t * c_{t-1} + i_t * \bar{c}_t$$

$$h_t = \bar{y}_t * \tanh(c_t)$$

Observation in LSTM and RNN

- Does LSTM solve our problem?
  - Vanishing gradient
  - Longer dependency
  - Parallel computation
- How can we overcome these?
- Self-attention model - transformer

Self-Attention network: transformer

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

Text normalization

- Three parts:
  - Words tokenization
  - Normalising word formats
  - Sentence segmentation

Word tokenization

- Segmenting running text into set of words
  - Based on white space
- Not only words:
  - M.Sc
  - RKMVERI
  - Dates: 17-10-2024
  - URL: https://rkmvu.ac.in/
  - Email: soumitra.samanta@gm.rkmvu.ac.in
  - Numbers: 100, 500.50
  - Clitic: We’re -> We are
  -

Subword tokenization

- Problems in word tokenisation ?
  - Unknown words in the test set
- Review, reviewer, low, lower, play, playing
  - Review, er, low, play, ing
- Many algorithms:
  - WordPiece[1]
  - Byte-pair encoding[2]
  - Unigram language modelling[3]
  - …
- Tokenization has two parts:
  - Token learner: convert a raw corpora text into tokens (vocabulary)
  - Token segmenter: convert a raw test sentence into the tokens in the vocabulary

[1]Achuster and Nakajima, Japanese and Korean voice search, In ICASSP, 2012
[2]Sennrich et al., Neural Machine Translation of Rare Words with Subword Units, In ACL 2016
[3]Kudo, Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates, In ACL, 2018

Byte-pair encoding (BPE)[1]

- Subwords
  - Review, reviewer, low, lower, play, playing
    - Review, er, low, play, ing

corpus	
5	l o w _
2	l o w e s t _
6	n e w e r _
3	w i d e r _
2	n e w _

vocabulary

_, d, e, i, l, n, o, r, s, t, w

'low' appears 5 times; 'lowest' 2 times and so on

[1]Sennrich et al., Neural Machine Translation of Rare Words with Subword Units, In ACL 2016
Example: Jurafsky & Martin "Speech and Language Processing, 3rd ed., 2023

Byte-pair encoding (BPE)[1]

- Merge most frequent pair (e and r 9 times)

corpus		vocabulary
5	l o w _	_, d, e, i, l, n, o, r, s, t, w, er
2	l o w e s t _	
6	n e w er _	
3	w i d er _	
2	n e w _	

[1]Sennrich et al., Neural Machine Translation of Rare Words with Subword Units, In ACL 2016
Example: Jurafsky & Martin "Speech and Language Processing, 3rd ed., 2023

Byte-pair encoding (BPE)[1]

- Merge next most frequent pair (er and _ 9 times)

corpus	
5	l o w _
2	l o w e s t _
6	n e w er_
3	w i d er_
2	n e w _

vocabulary

_, d, e, i, l, n, o, r, s, t, w, er, er_

[1]Sennrich et al., Neural Machine Translation of Rare Words with Subword Units, In ACL 2016
Example: Jurafsky & Martin "Speech and Language Processing, 3rd ed., 2023

Byte-pair encoding (BPE)[1]

- Merge next most frequent pair (n and e 8 times)

corpus		vocabulary
5	l o w _	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne
2	l o w e s t _	
6	ne w er_	
3	w i d er_	
2	ne w _	

[1]Sennrich et al., Neural Machine Translation of Rare Words with Subword Units, In ACL 2016
Example: Jurafsky & Martin "Speech and Language Processing, 3rd ed., 2023

Byte-pair encoding (BPE)[1]

- Continue merging unto to a desired number of new tokens

merge	current vocabulary
(ne, w)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new
(l, o)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo
(lo, w)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo, low
(new, er_)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo, low, newer_
(low, _)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo, low, newer_, low_

[1]Sennrich et al., Neural Machine Translation of Rare Words with Subword Units, In ACL 2016
Example: Jurafsky & Martin "Speech and Language Processing, 3rd ed., 2023

Byte-pair encoding (BPE)[1]

- Formal algorithm

function BYTE-PAIR ENCODING(strings C, number of merges k) **returns** vocab V

$V \leftarrow$ all unique characters in C # initial set of tokens is characters
for $i = 1$ **to** k **do** # merge tokens k times
  $t_L, t_R \leftarrow$ Most frequent pair of adjacent tokens in C
  $t_{NEW} \leftarrow t_L + t_R$ # make new token by concatenating
  $V \leftarrow V + t_{NEW}$ # update the vocabulary
  Replace each occurrence of t_L, t_R in C with t_{NEW} # and update the corpus
return V

[1]Sennrich et al., Neural Machine Translation of Rare Words with Subword Units, In ACL 2016
Example: Jurafsky & Martin "Speech and Language Processing, 3rd ed., 2023

Word normalization

- Case folding:
  - Uh-huh vs uhhhh
  - USA vs US
- Lemmatization - two words have the same root, despite their surface difference
  - Am, is , are - be
  - Cat, cats - cat
  - …
- How can we lemmatise a word?
  - Morphological analysis
  - Morphology - study of the way words are built up from smaller meaning-bearing unit: morphemes
    - Two types
      - Stems - central morpheme, responsible for the main meaning
      - Affixes - adding additional meaning
    - Example: cats - cat and s

Lemmatization

- Stemming - chopping off word-final affixes
- Porter stemmer (Martin F. Porter, 1980)
  - Rule based - set of rules:

```
SSES -> SS (caresses -> caress)
IES  -> I (ties -> ti)
ATIONAL -> ATE (relational ->relate
ING -> ϵ (playing -> play)
```

```
Step 1a

    SSES -> SS                         caresses  ->  caress
    IES  -> I                          ponies    ->  poni
                                       ties      ->  ti
    SS   -> SS                         caress    ->  caress
    S    ->                            cats      ->  cat

Step 1b

    (m>0) EED -> EE                    feed      ->  feed
                                       agreed    ->  agree
    (*v*) ED  ->                       plastered ->  plaster
                                       bled      ->  bled
    (*v*) ING ->                       motoring  ->  motor
                                       sing      ->  sing

If the second or third of the rules in Step 1b is successful, the following
is done:

    AT -> ATE                       conflat(ed)  ->  conflate
    BL -> BLE                       troubl(ed)   ->  trouble
    IZ -> IZE                       siz(ed)      ->  size
    (*d and not (*L or *S or *Z))
       -> single letter
                                    hopp(ing)    ->  hop
                                    tann(ed)     ->  tan
                                    fall(ing)    ->  fall
                                    hiss(ing)    ->  hiss
                                    fizz(ed)     ->  fizz
    (m=1 and *o) -> E               fail(ing)    ->  fail
                                    fil(ing)     ->  file
```

Jurafsky & Martin “Speech and Language Processing, 3rd ed., 2023

Porter stemmer

- Detail rules: https://tartarus.org/martin/PorterStemmer/def.txt

```
Step 1c

    (*v*) Y -> I                    happy        ->  happi
                                    sky          ->  sky

Step 1 deals with plurals and past participles. The subsequent steps are
much more straightforward.

Step 2

    (m>0) ATIONAL ->  ATE           relational     ->  relate
    (m>0) TIONAL  ->  TION          conditional    ->  condition
                                    rational       ->  rational
    (m>0) ENCI    ->  ENCE          valenci        ->  valence
    (m>0) ANCI    ->  ANCE          hesitanci      ->  hesitance
    (m>0) IZER    ->  IZE           digitizer      ->  digitize
    (m>0) ABLI    ->  ABLE          conformabli    ->  conformable
    (m>0) ALLI    ->  AL            radicalli      ->  radical
    (m>0) ENTLI   ->  ENT           differentli    ->  different
    (m>0) ELI     ->  E             vileli        - >  vile
    (m>0) OUSLI   ->  OUS           analogousli    ->  analogous
    (m>0) IZATION ->  IZE           vietnamization ->  vietnamize
    (m>0) ATION   ->  ATE           predication    ->  predicate
    (m>0) ATOR    ->  ATE           operator       ->  operate
    (m>0) ALISM   ->  AL            feudalism      ->  feudal
    (m>0) IVENESS ->  IVE           decisiveness   ->  decisive
    (m>0) FULNESS ->  FUL           hopefulness    ->  hopeful
    (m>0) OUSNESS ->  OUS           callousness    ->  callous
    (m>0) ALITI   ->  AL            formaliti      ->  formal
    (m>0) IVITI   ->  IVE           sensitiviti    ->  sensitive
    (m>0) BILITI  ->  BLE           sensibiliti    ->  sensible
```

```
Step 3

    (m>0) ICATE ->  IC              triplicate     ->  triplic
    (m>0) ATIVE ->                  formative      ->  form
    (m>0) ALIZE ->  AL              formalize      ->  formal
    (m>0) ICITI ->  IC              electriciti    ->  electric
    (m>0) ICAL  ->  IC              electrical     ->  electric
    (m>0) FUL   ->                  hopeful        ->  hope
    (m>0) NESS  ->                  goodness       ->  good

Step 4

    (m>1) AL    ->                  revival        ->  reviv
    (m>1) ANCE  ->                  allowance      ->  allow
    (m>1) ENCE  ->                  inference      ->  infer
    (m>1) ER    ->                  airliner       ->  airlin
    (m>1) IC    ->                  gyroscopic     ->  gyroscop
    (m>1) ABLE  ->                  adjustable     ->  adjust
    (m>1) IBLE  ->                  defensible     ->  defens
    (m>1) ANT   ->                  irritant       ->  irrit
    (m>1) EMENT ->                  replacement    ->  replac
    (m>1) MENT  ->                  adjustment     ->  adjust
    (m>1) ENT   ->                  dependent      ->  depend
    (m>1 and (*S or *T)) ION ->     adoption       ->  adopt
    (m>1) OU    ->                  homologou      ->  homolog
    (m>1) ISM   ->                  communism      ->  commun
    (m>1) ATE   ->                  activate       ->  activ
    (m>1) ITI   ->                  angulariti     ->  angular
    (m>1) OUS   ->                  homologous     ->  homolog
    (m>1) IVE   ->                  effective      ->  effect
    (m>1) IZE   ->                  bowdlerize     ->  bowdler
```

Sentence segmentation

- Segment the running text into sentences
- How?
- Punctuation
  - Period/full stop(.) - ambiguous (R.K.M.V.E.R.I, .com, Inc., 23.45)
  - Question mark(?) - unambiguous
  - exclamation(!) - unambiguous
- How can we handle ambiguity?
  - Sentence and word segmentation should be done jointly
  - Decide the period is a part of the word or not
    - Used a abbreviation dictionary

Edit distance

- The word student may be misspell as stdent
  - How can we correct ?
  - Not only in word level but it sentence level also
    - In RKMVERI there is a course on Big Data
    - In RKMVERI there is a course on Big Data Analytics
- Edit distance quantify the similarity between two strings by
  - Addition - stdent -> student
  - Deleting - stuudent -> student
  - Substitution - studant -> student
- Minimum edit distance - minimum number operations required to transform one string to another string
- How can we find the minimum edit distance between two strings?

Minimum edit distance (MED) algorithm

- Dynamic programming based proposed by Wagner and Fischer, 1974
  - Source string S_1 of length m
  - Target string S_2 of length n
  - Define a matrix D, such that $D[i, j]$ gives the edit distance between $S_1[1..i]$ and $S_2[1..j]$
  - What about $D[i,0]$ and $D[0,j]$?
  - $$D[i, j] = min \begin{cases} D[i-1,j] + del-cost(S_1[i]) \\ D[i, j-1] + ins-cost(S_2[j]) \\ D[i-1,j-1] + sub-cost(S_1[i], S_2[j]) \end{cases}$$
  - Define cost: $del - cost(S[i]) = 1$, $ins - cost(S[i]) = 1$, $sub - cost(S_1[i], S_2[j]) = 2$?

Jurafsky & Martin "Speech and Language Processing, 3rd ed., 2023

MED algorithm

```
function MIN-EDIT-DISTANCE(source, target) returns min-distance

  n ← LENGTH(source)
  m ← LENGTH(target)
  Create a distance matrix D[n+1,m+1]

  # Initialization: the zeroth row and column is the distance from the empty string
  D[0,0] = 0
  for each row i from 1 to n do
      D[i,0] ← D[i-1,0] + del-cost(source[i])
  for each column j from 1 to m do
      D[0,j] ← D[0, j-1] + ins-cost(target[j])

  # Recurrence relation:
  for each row i from 1 to n do
        for each column j from 1 to m do
           D[i, j] ← MIN( D[i−1, j] + del-cost(source[i]),
                          D[i−1, j−1] + sub-cost(source[i], target[j]),
                          D[i, j−1] + ins-cost(target[j]))
  # Termination
  return D[n,m]
```

Jurafsky & Martin "Speech and Language Processing, 3rd ed., 2023

Example: MED

- Source: $S_1 = intention$ and Target: $S_2 = execution$

Src\Tar	#	e	x	e	c	u	t	i	o	n
#	0	1	2	3	4	5	6	7	8	9
i	1	2	3	4	5	6	7	6	7	8
n	2	3	4	5	6	7	8	7	8	7
t	3	4	5	6	7	8	7	8	9	8
e	4	3	4	5	6	7	8	9	10	9
n	5	4	5	6	7	8	9	10	11	10
t	6	5	6	7	8	9	8	9	10	11
i	7	6	7	8	9	10	9	8	9	10
o	8	7	8	9	10	11	10	9	8	9
n	9	8	9	10	11	12	11	10	9	8

- What about stdent -> student ?

Jurafsky & Martin “Speech and Language Processing, 3rd ed., 2023

String alignment

- Source: $S_1 = intention$ and Target: $S_2 = execution$

	#	e	x	e	c	u	t	i	o	n
#	0	← 1	← 2	← 3	← 4	← 5	← 6	← 7	← 8	← 9
i	↑ **1**	↖←↑ 2	↖←↑ 3	↖←↑ 4	↖←↑ 5	↖←↑ 6	↖←↑ 7	↖ 6	← 7	← 8
n	↑ 2	↖←↑ **3**	↖←↑ 4	↖←↑ 5	↖←↑ 6	↖←↑ 7	↖←↑ 8	↑ 7	↖←↑ 8	↖ 7
t	↑ 3	↖←↑ 4	↖←↑ **5**	↖←↑ 6	↖←↑ 7	↖←↑ 8	↖ 7	←↑ 8	↖←↑ 9	↑ 8
e	↑ 4	↖ 3	← 4	↖← **5**	← **6**	← 7	←↑ 8	↖←↑ 9	↖←↑ 10	↑ 9
n	↑ 5	↑ 4	↖←↑ 5	↖←↑ 6	↖←↑ 7	↖←↑ **8**	↖←↑ 9	↖←↑ 10	↖←↑ 11	↖↑ 10
t	↑ 6	↑ 5	↖←↑ 6	↖←↑ 7	↖←↑ 8	↖←↑ 9	↖ **8**	← 9	← 10	←↑ 11
i	↑ 7	↑ 6	↖←↑ 7	↖←↑ 8	↖←↑ 9	↖←↑ 10	↑ 9	↖ **8**	← 9	← 10
o	↑ 8	↑ 7	↖←↑ 8	↖←↑ 9	↖←↑ 10	↖←↑ 11	↑ 10	↑ 9	↖ **8**	← 9
n	↑ 9	↑ 8	↖←↑ 9	↖←↑ 10	↖←↑ 11	↖←↑ 12	↑ 11	↑ 10	↑ 9	↖ **8**

Jurafsky & Martin "Speech and Language Processing, 3rd ed., 2023