Midterm stuffs

Misspelling detection and correction

KBTG-CU-NECTEC NLP

KBTG จับมือ NECTEC และจุฬาฯ เปิดตัว Thai NLP สร้างอีโคซิสเต็มพัฒนา NLP ภาษาไทย

By: sponsored on 16 December 2019 - 12:24 Tags: Kasikorn Bank NECTEC Artificial Intelligence Natural Language Thailand Advertorial

KBTG หรือ KASIKORN Business-Technology Group ประกาศความร่วมมือกับ ศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ (NECTEC ในสังกัด สวทช.) และจุฬาลงกรณ์มหาวิทยาลัย เปิดตัวโครงการ Thai NLP ร่วมพัฒนาเทคโนโลยีประมวลผลภาษาธรรมชาติของภาษาไทย โดยเปิดเป็น Open API และตั้งเป้าสร้างและพัฒนาระบบนิเวศน์ ไปจนถึงชุมชนของ NLP ภาษาไทยขึ้นมาเอง โดยมีคุณชัตติยา อินทรวิชัย กรรมการผู้จัดการ ธนาคารกสิกรไทยมาเป็นประธานในการเปิดตัว NLP

https://www.blognone.com/node/113694

Leaderboard

TA01	7.54	7.5119
TA02	3.37	3.6804

ent ID	Public test CER	Private test CER
97521	2.62	2.615
85221	2.65	2.646
41021	2.48	2.6512
54421	2.6	2.6626
19921	2.69	2.6688
24821	2.64	2.6688
42121	2.68	2.7381
50121	2.68	2.7598
04021	2.84	2.7712
03221	2.82	2.8022
68021	2.78	2.8074
26521	2.8	2.9171
44521	2.83	2.9233
82211	2.78	3.0174
30803	2.98	3.0329
21021	3	3.0412

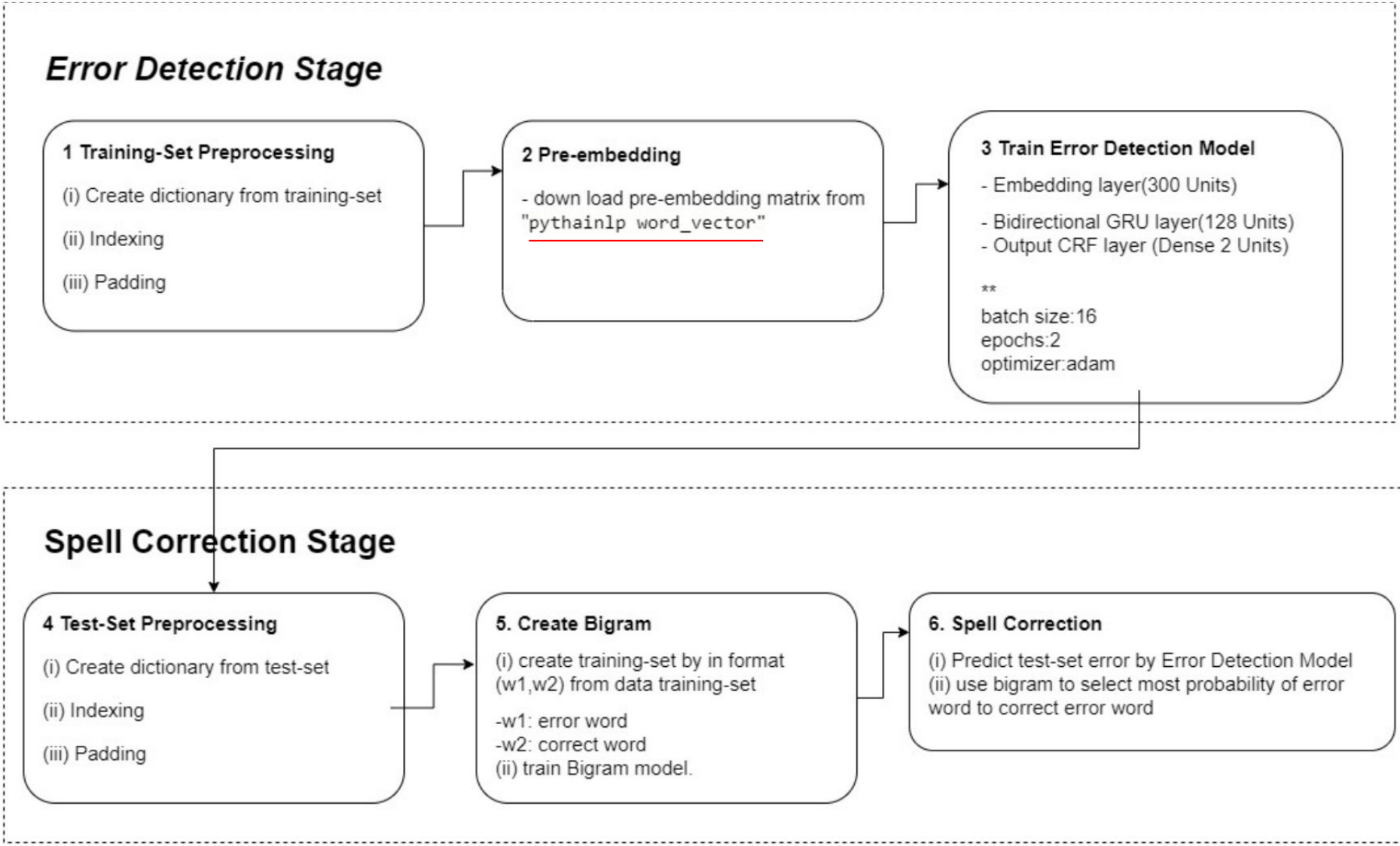
Weak baseline

Rule-based CER 5.01

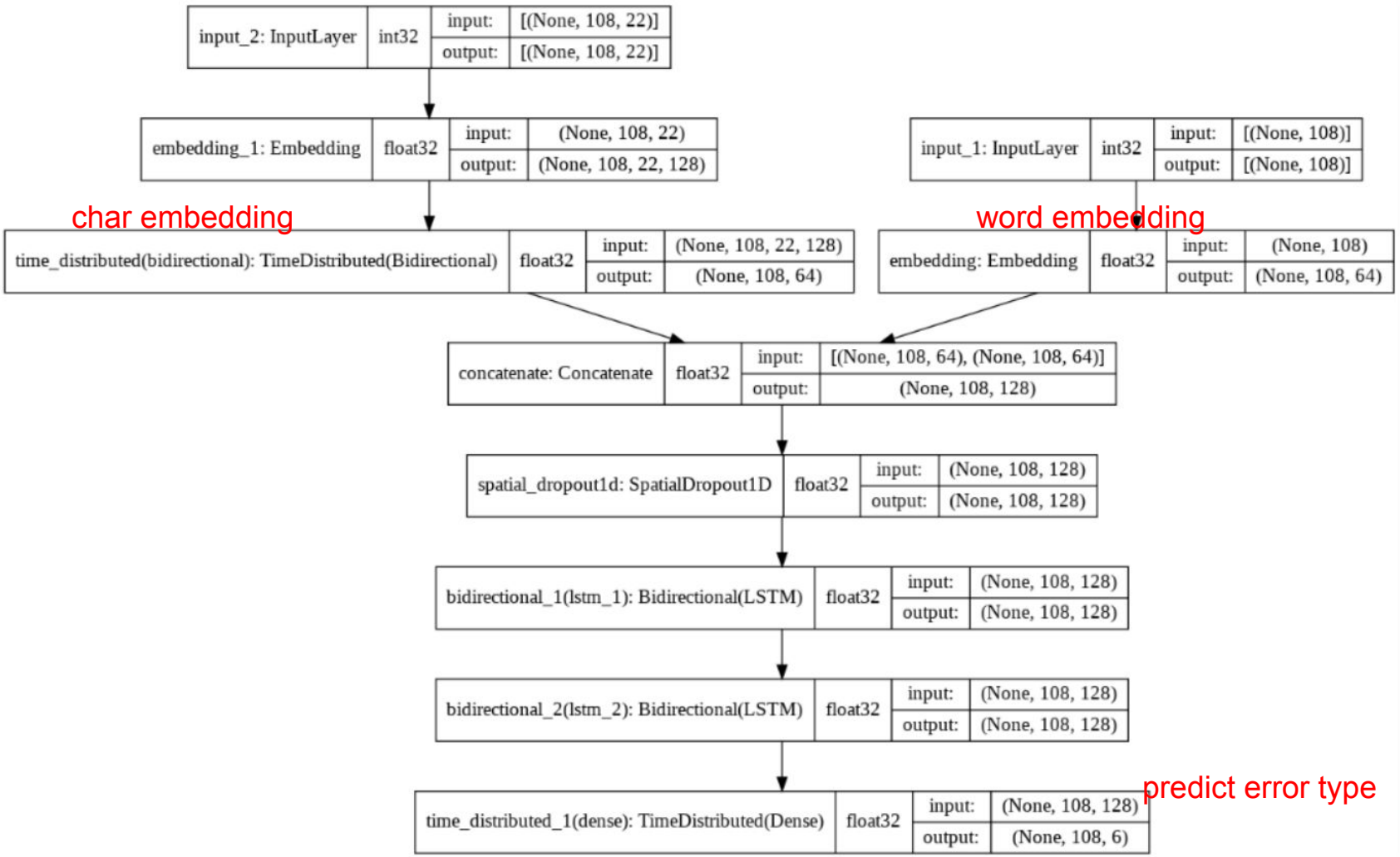
Every unique correction gets corrected

หริอ หรือ
ยังงัย ยังไง
พัชชาา พัชชา
คะ ค่ะ
เบอ เบอร์
โทรสับ โทรศัพท์
เค้า เขา
ดิ สิ
เจ้าของ คอมเมนต์
เชค เช็ค
ใหม ไหม
ค่ะ คะ
ไ ทยพาณิช ไทยพาณิชย์
ม๊ัย ไหม
ชูวิท ชูวิทย์
ด่อก ดอก
อ่ะ อะ
อะ อ่ะ
ตีมมิ่งล่ม สตรีมมิ่งล่ม
พันธิป พันทิป
สอบถามึะ สอบถาม
ก้ ก็
ฮอลิวู้ด ฮอลลีวูด
ตรึม เต็ม
เก๊า เค้า
ฮืออออ ฮือ
โน เงิน
กะ กับ

3rd


Error Detection Stage
1 Training-Set Preprocessing
(i) Create dictionary from training-set
(ii) Indexing
(iii) Padding
2 Pre-embedding
- down load pre-embedding matrix from "pythainlp word_vector"
3 Train Error Detection Model
- Embedding layer(300 Units)
- Bidirectional GRU layer(128 Units)
- Output CRF layer (Dense 2 Units)
**
batch size:16
epochs:2
optimizer:adam
Spell Correction Stage
4 Test-Set Preprocessing
(i) Create dictionary from test-set
(ii) Indexing
(iii) Padding
5. Create Bigram
(i) create training-set by in format (w1,w2) from data training-set
-w1: error word
-w2: correct word
(ii) train Bigram model.
6. Spell Correction
(i) Predict test-set error by Error Detection Model
(ii) use bigram to select most probability of error word to correct error word

2nd detection stage



2nd correction stage

For consequent misspells, merge tokens

Set of rules

1) Pairs of error-correction, if multiple, use most frequent correction

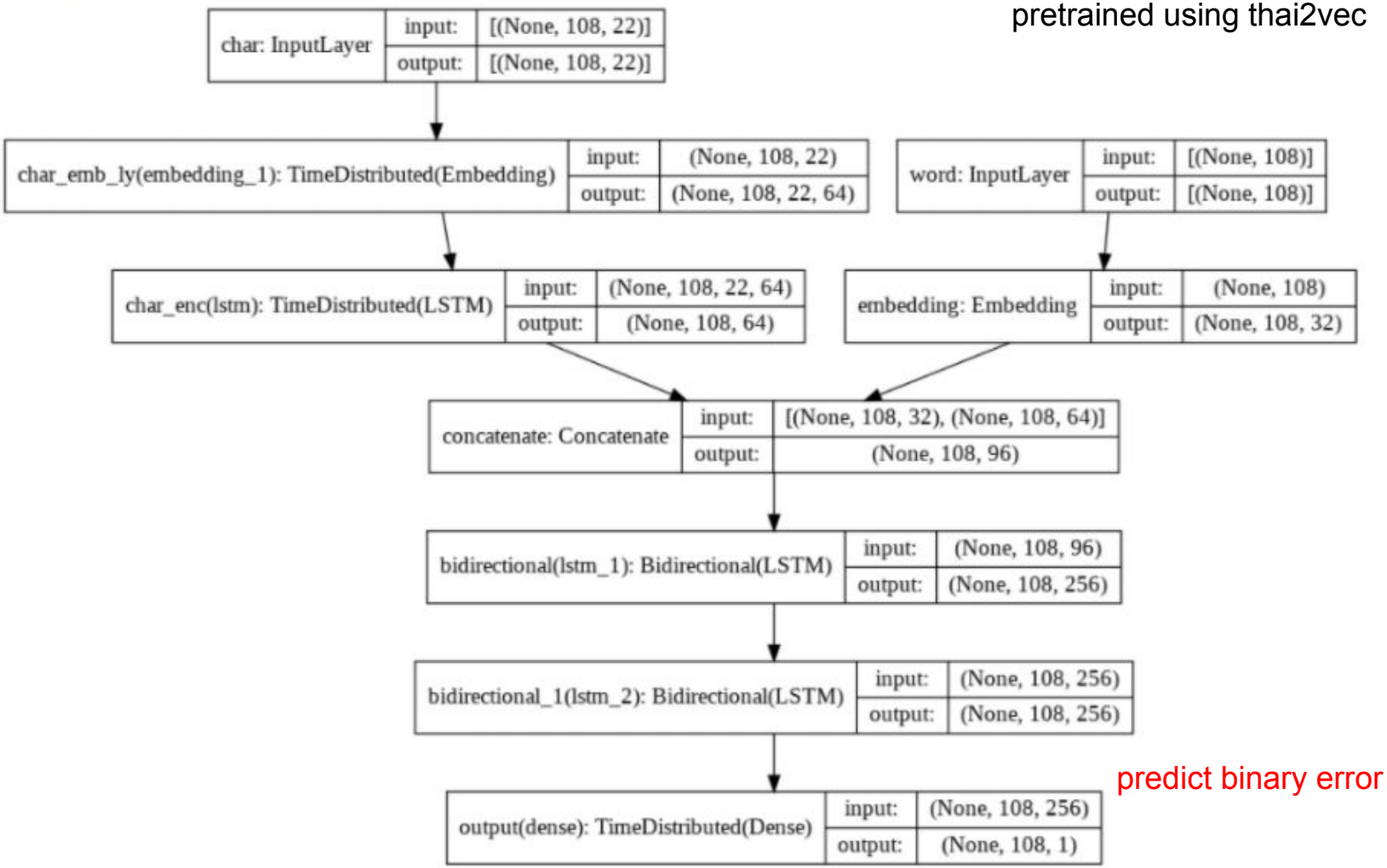
2) Simple rules

3) For morph errors, use word segment to remove the extra characters (แล้วววววว -> แล้ว วววววว)

```
final_final_list = []
for line in final_list:
  temp = []
  temp.append(line[0])
  for i in range(1, len(line)):
    if line[i] == 'ค่ะ' and line[i - 1] == 'ไหม':
      temp.append("คะ")
    elif line[i] == 'ค่ะ' and line[i - 1] == 'ไง':
      temp.append("คะ")
    elif line[i] == 'ค่ะ' and line[i - 1] == 'นะ':
      temp.append("คะ")
    elif line[i] == 'คะ' and line[i - 1] == 'เลย':
      temp.append("ค่ะ")
    elif line[i] == 'คะ' and line[i - 1] == 'แล้ว':
      temp.append("ค่ะ")
    elif line[i] == 'คะ' and line[i - 1] == 'หน่อย':
      temp.append("ค่ะ")
    else:
      temp.append(line[i])
```

1st detection stage

pretrained using thai2vec



predict binary error

1st correction stage

For consequent misspells, merge tokens

Pairs of error-correction, if multiple, use most frequent correction

Honorable mention 6th

Pure rulebased

Learn misspell pairs including bigram pairs to handle ค่ะ คะ

(w_{n-1}, w_n, c_n)

Since there are no detector, correct pairs is also learned to counter real word mispellings

Lots of processing

- merge duplicate characters
- remove punctuations
- etc

Notes

Top rankers use rulebase on the correction stage.

Some submission use a very elaborate correction process such as using BERT. However, the data is probably too small.

Cons of rule based?

TA02

Detection then correction

Detection using only word feature

- Predict mispelt/correct

```
def get_detection_model():
  inp = tf.keras.layers.Input((120,))
  em = tf.keras.layers.Embedding(len(input_dict), 300,  input_length=120, mask_zero=True)(inp)
  x = tf.keras.layers.Bidirectional(tf.keras.layers.GRU(64, return_sequences=True))(em)
  x = tf.keras.layers.Bidirectional(tf.keras.layers.GRU(64, return_sequences=True))(x)
  # x = tf.keras.layers.Concatenate()([x, em])
  x = tf.keras.layers.TimeDistributed( tf.keras.layers.Dense(1, activation = 'sigmoid'))(x)
  model = tf.keras.models.Model(inputs = inp, outputs = x)
  return model
```

TA02

Detection then correction

Correction used binarized prediction from detection step as additional input.

Predict output word

```
def get_correction_model():
  inp = tf.keras.layers.Input((120,))
  em = tf.keras.layers.Embedding(len(input_dict), 300,  input_length=120, mask_zero=True)(inp)
  x = tf.keras.layers.Bidirectional(tf.keras.layers.GRU(64, return_sequences=True))(em)
  x = tf.keras.layers.Bidirectional(tf.keras.layers.GRU(64, return_sequences=True))(x)
  inp2 = tf.keras.layers.Input((120, 1))
  x = tf.keras.layers.Concatenate()([x, inp2])
  x = tf.keras.layers.TimeDistributed( tf.keras.layers.Dense(len(input_dict), activation = 'softmax'))(x)
  model = tf.keras.models.Model(inputs = [inp, inp2], outputs = x)
  return model
```

Error propagation and NLP systems

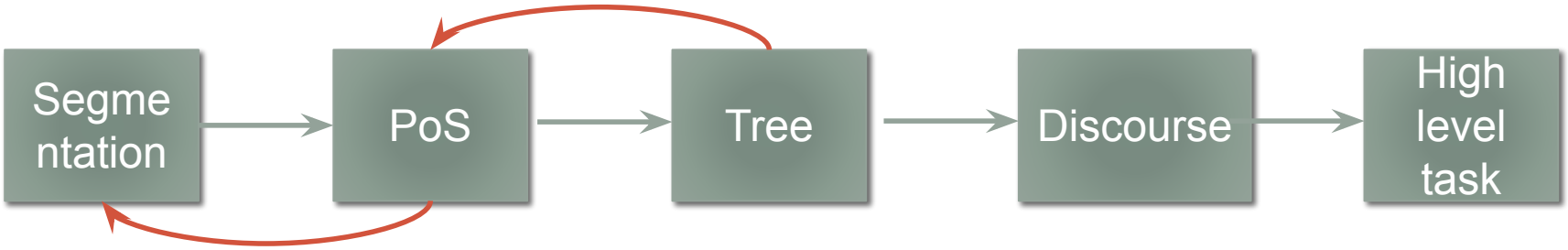
- Typical flow of NLP systems

เขาต้องมีหมอนกอด vs เขาต้องมีหมอรกอด

Pipeline analysis

Error propagation and NLP systems

- Use higher level info to help lower level
  - Iterative process



Joint analysis

Error propagation and NLP systems

- Do the task together

Joint analysis

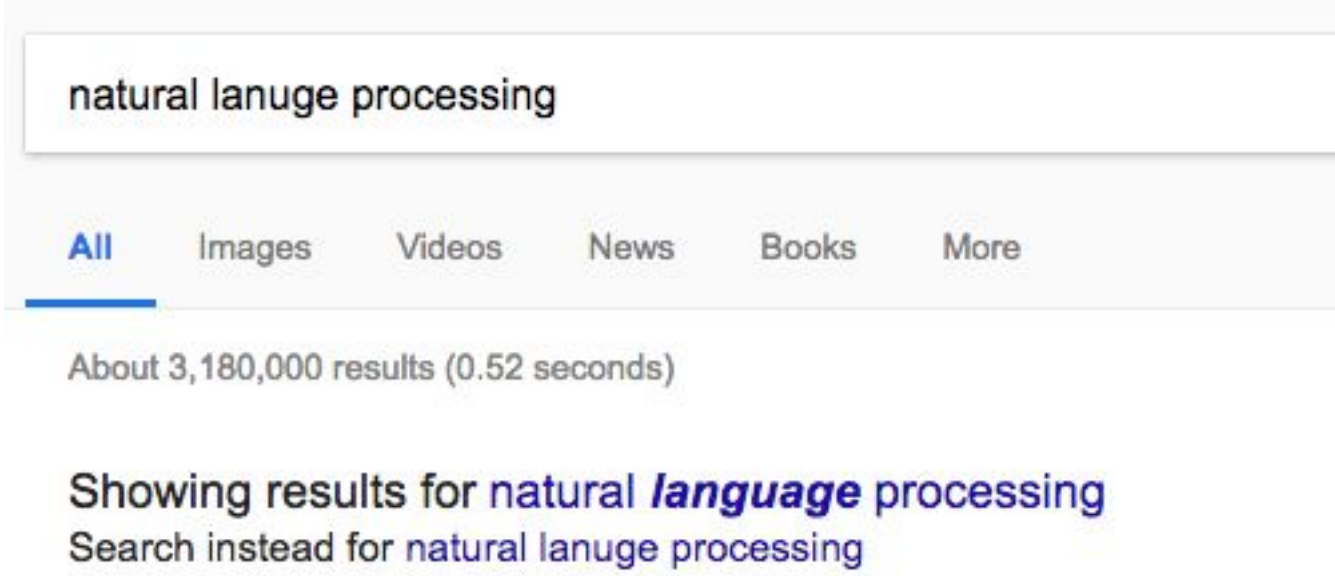
Error propagation and NLP systems

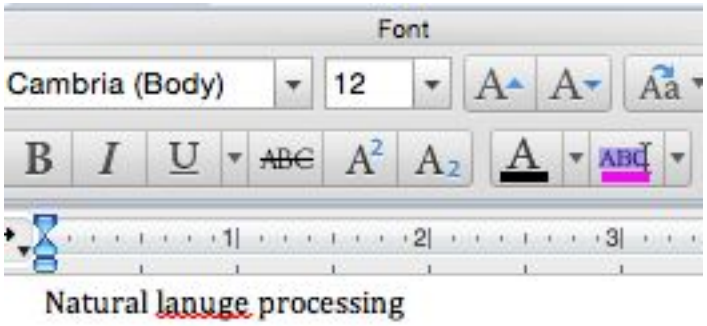
- One single task
  - Input closest to the input form – characters, words

High level task with low level structures as input

End-to-end analysis

Spell correction



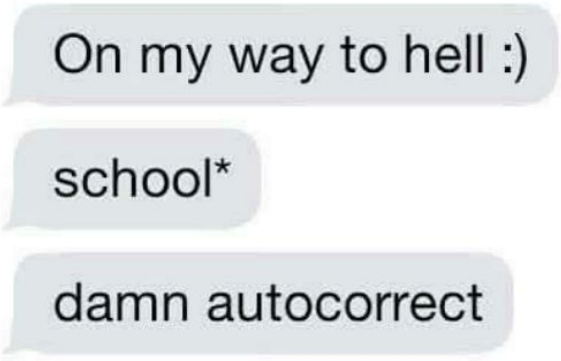


Rates of spelling errors ~1-20% depending on application

Tasks

- Spell error detection
- Spell error correction
  - Suggestion list
  - Suggest a correction
  - Autocorrect

Confidence of the system



Type of spelling errors

- Non-word errors
  - Languaeg -> Language
- Real-word errors
  - Typographical errors
    - Three -> there
    - เขาต้องมีหมอรกอด
  - Cognitive errors (homophones)
    - piece->peace
    - too->two
    - ค่ะ -> คะ

- Dictionary is required.

Corpora

- What kind of text?
  - Pantip, twitter, wikipedia, textbook, …
- Extra: How to actually construct the training corpora?
  - Look at standard sentences and add misspellings
    - By keyboard distance, etc.

Possible solutions

Detecting errors:

- Segmentation failure (word not in dictionary)
- Using prediction model

Correcting errors:

- Simple N-grams
- CBOW (neural based)
- Edit distance

End-to-end

- Seq2seq model (machine translation-based)

A note on dictionary

- Require a large dictionary (millions) to handle real world internet usage
- Automatically generated from unigrams list
  - This list includes misspellings
  - Misspellings occur less frequently than correct words in similar context
- Or let users add new words when not accepting corrections
- Cute solution using NER to ignore NE words if OOV

Whitelaw, C., Hutchinson, B., Chung, G. Y., and Ellis,G. (2009). Using the web for language independent spellchecking and autocorrection

Assumptions

- This only handle one misspelled word per sentence?
  - How to handle multiple missing words in a row?
- Word level correction:
  - Cannot correct OOV words
- Character level correction:
  - Detecting misspellings in the character level might not be able to handle deletion errors

ฉ ั น ไ ป เ ท ี ้ ย ว

0 0 0 0 0 0 0 0 1 0 0

ฉ ั น ไ ป เ ท ี ย ว

0 0 0 0 0 0 0 0 0 0 0

Assumptions

- Discourse?
  - Only a couple of students actually talks about discourse

A: เดี๋ยวพรุ่งนี้เราต้องไปที่น่าน
B: จะไปที่นะนทำไมละ

นนท์ vs น่าน vs นั่น

Spell correction in literature

1. Non-word
2. Real-word

Non-word spelling error

- Detection:
  - Any word not in dictionary is an error
- Correction:
  1. Generate candidates
  2. Choose the one that has the best
     1. Smallest edit distance
     2. Best noisy channel probability

Real word spelling error

- Detection:
  - ???
- Correction:
  1. Generate candidates
     1. From similar pronunciations
     2. From similar spellings
  2. Choose the one that has the best
     1. Smallest edit distance
     2. Best noisy channel probability

  The criterions has to be context sensitive!

  ไปไหนดีค่ะ -> ไปไหนดีคะ

Candidate generation

- Words with similar spelling
  - Edit distance
  - Typically at most 2 edit distance
  - Can be weighted based on the task, e.g. keyboard
  - Allow merging and splitting of words
    - Ladygaga -> Lady Gaga
    - |การะ|เกด| -> การะเกด
- Words with similar pronunciation
  - Example: Aspell, Soundex – a hash based on sounds
    - Jurafsky, Jarofsky, Jarovsky, and Jarovski all map to J612
  - Modern methods use Letter-2-sound (L2S) rules (also called Grapheme-to-Phoneme G2P)
- How to generate this in an efficient manner?

Candidate generation

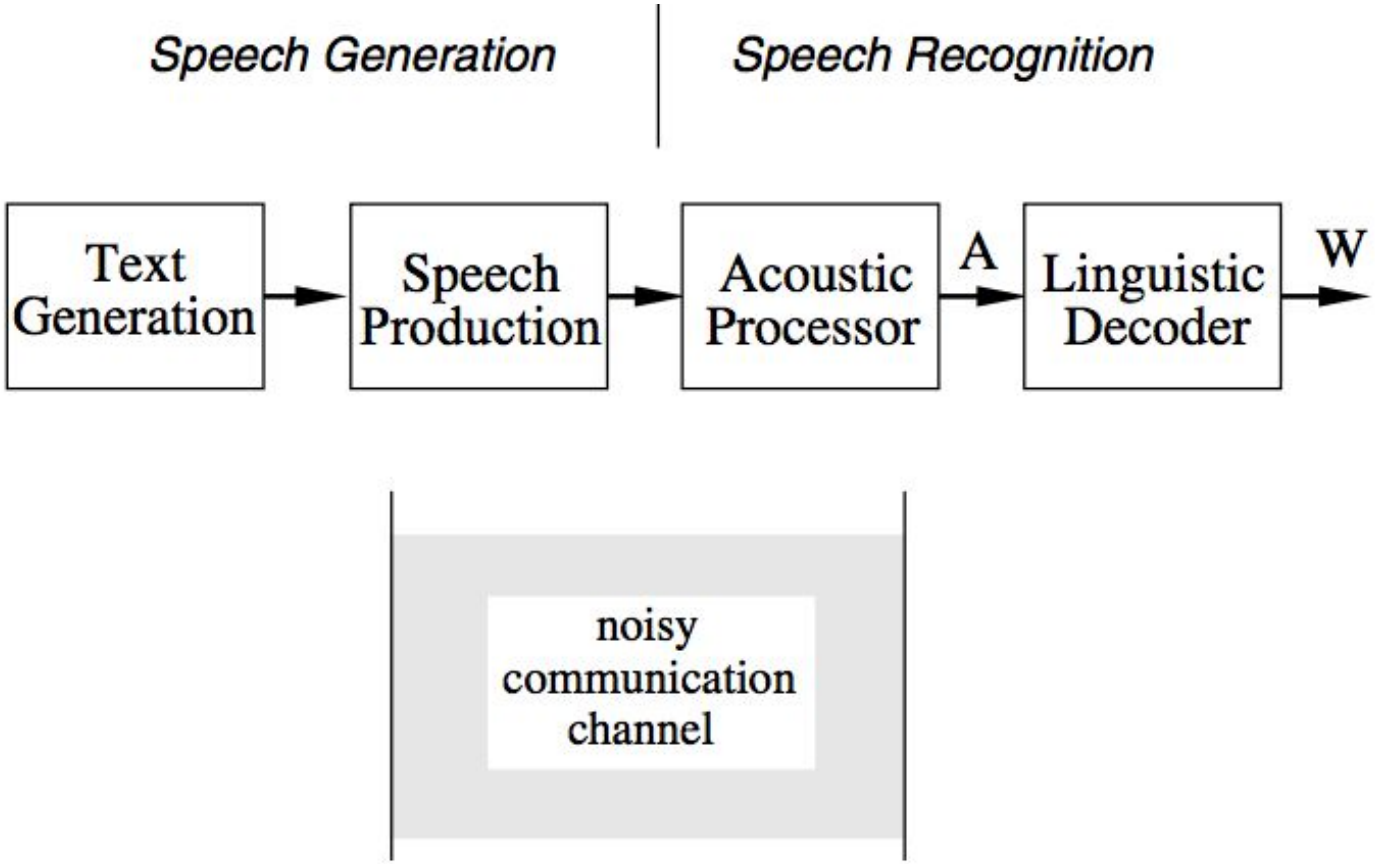
1. Run through dictionary, check edit distance (slow)
2. Generate all words with desired edit distance, then intersect with dictionary
3. Character n-gram inverted index. Find words which shared the most n-gram (fast)
   - Google -> goo oog ogl gle
   - Inverted index: goo -> google good goon good-bye
4. Compute with Finite State Transducer (FST) (fast)
5. Precomputed map of words
6. Mixture (https://github.com/wolfgarbe/SymSpell) (fast)

Tommi A Pirinen. Weighted Finite-State Methods for Spell-Checking and Correction 2014

The noisy channel model

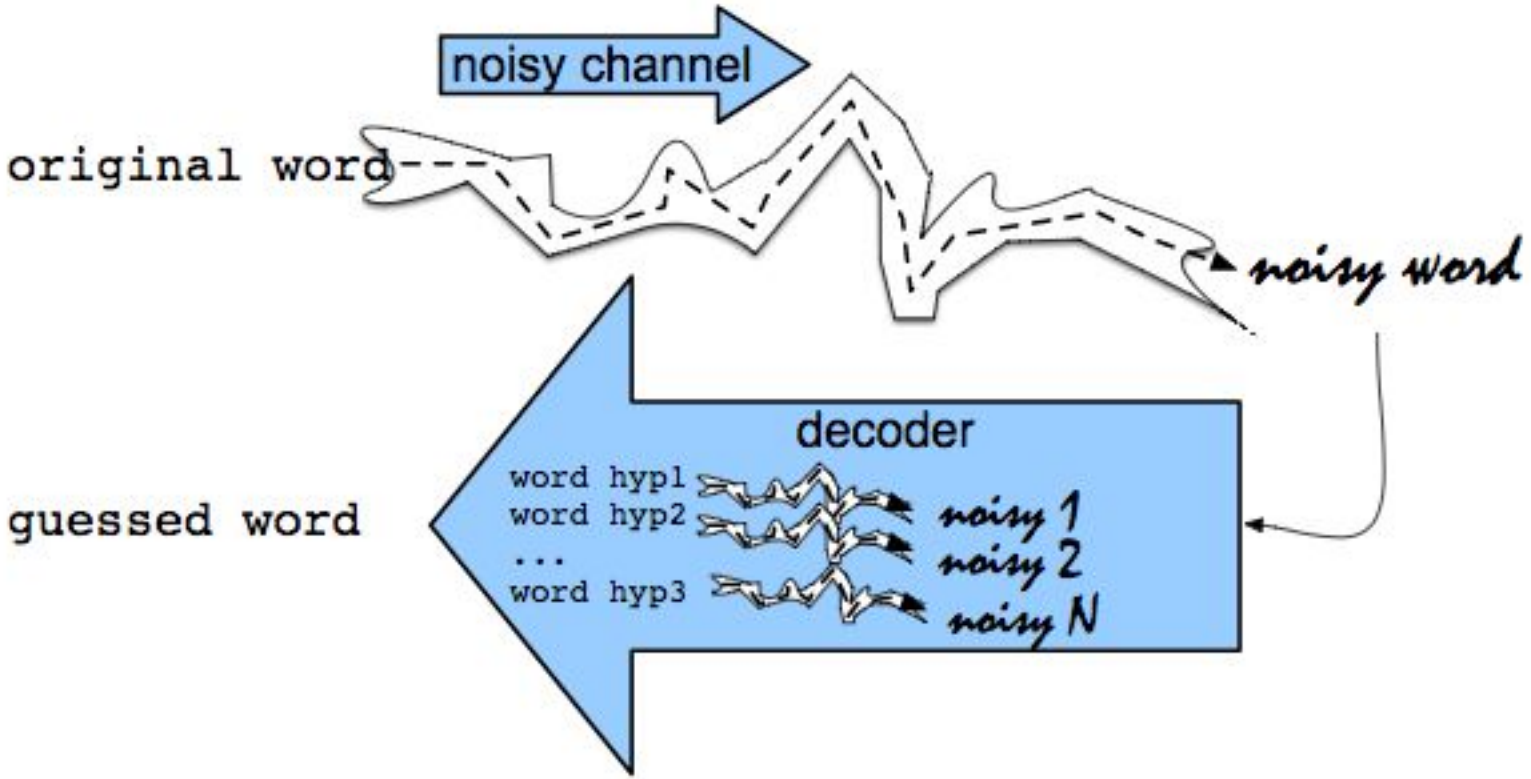
- Used to rank the best candidates

Information theoretic formulation



$$W^* = \arg\max_W P(W \mid A) \qquad P(W \mid A) = \frac{P(A \mid W)P(W)}{P(A)}$$

The noisy channel intuition



Jurafsky, chapter 5

Bayes rule

- An observation x of the misspelled word
- Find the correct word w* from vocab list V

$$= argmax_{w \in V} P(w|x)$$

$$= argmax_{w \in V} \frac{P(x|w)P(w)}{P(x)}$$

Language model

$$= argmax_{w \in V} P(x|w)P(w)$$

Channel/error model

Noisy channel example

- acress
- Let’s generate candidates

Noisy channel example

Edit distance 1 (Damerau-Levenshtein distance)

Levenshtein edit distance – insertion, deletion, substitution
Damerau-Levenshtein edit distance - insertion, deletion, substitution, transposition

		Transformation			
Error	Correction	Correct Letter	Error Letter	Position (Letter #)	Type
acress	actress	t	—	2	deletion
acress	cress	—	a	0	insertion
acress	caress	ca	ac	0	transposition
acress	access	c	r	2	substitution
acress	across	o	e	3	substitution
acress	acres	—	s	5	insertion
acress	acres	—	s	4	insertion

Jurafsky, chapter 5

Language model (unigram)

w	count(w)	p(w)
actress	9,321	.0000231
cress	220	.000000544
caress	686	.00000170
access	37,038	.0000916
across	120,844	.000299
acres	12,874	.0000318

Channel/error model

- Easiest to use a confusion matrix conditioned on the previous character.
- Based on counts

$$P(x|w) = \begin{cases} \dfrac{\text{del}[x_{i-1}, w_i]}{\text{count}[x_{i-1} w_i]}, & \text{if deletion} \\ \dfrac{\text{ins}[x_{i-1}, w_i]}{\text{count}[w_{i-1}]}, & \text{if insertion} \\ \dfrac{\text{sub}[x_i, w_i]}{\text{count}[w_i]}, & \text{if substitution} \\ \dfrac{\text{trans}[w_i, w_{i+1}]}{\text{count}[w_i w_{i+1}]}, & \text{if transposition} \end{cases}$$

w_i is the i-th character

Learning the confusion matrix

1. Have a data of misspellings-corrections pair
2. Have a data of misspellings and use EM
    1. Initialize the confusion matrix to be equally likely
    2. E-step: run the spell checker and correct words
    3. M-step: re-estimate the confusion matrix based on corrections

Or use nearby keys
(Each phone need to have different model)

Channel/error model

Candidate Correction	Correct Letter	Error Letter	x|w	P(x|w)
actress	t	-	c|ct	.000117
cress	-	a	a|#	.00000144
caress	ca	ac	ac|ca	.00000164
access	c	r	r|c	.000000209
across	o	e	e|o	.0000093
acres	-	s	es|e	.0000321
acres	-	s	ss|s	.0000342

Noisy channel model

Candidate Correction	Correct Letter	Error Letter	x|w	P(x|w)	P(w)	10^9*P(x|w)P(w)
actress	t	-	c|ct	.000117	.0000231	2.7
cress	-	a	a|#	.00000144	.000000544	0.00078
caress	ca	ac	ac|ca	.00000164	.00000170	0.0028
access	c	r	r|c	.000000209	.0000916	0.019
across	o	e	e|o	.0000093	.000299	2.8
acres	-	s	es|e	.0000321	.0000318	1.0
acres	-	s	ss|s	.0000342	.0000318	1.0

Note1: It is typical to use more than unigrams for the LM (very important for real word errors)

Note2: typically added a weight parameter tuned on dev set

$$P(x|w)P(w)^{\alpha}$$

Real word case for noisy channel

- Need to compare candidates against no correction
  - Error model needs a score for no correction
- Set to some parameter, c, representing how errorful our input is

- $P(w|x) = c$ for $w = x$

- The rest of the probability 1-c is weighted by the confusion matrix

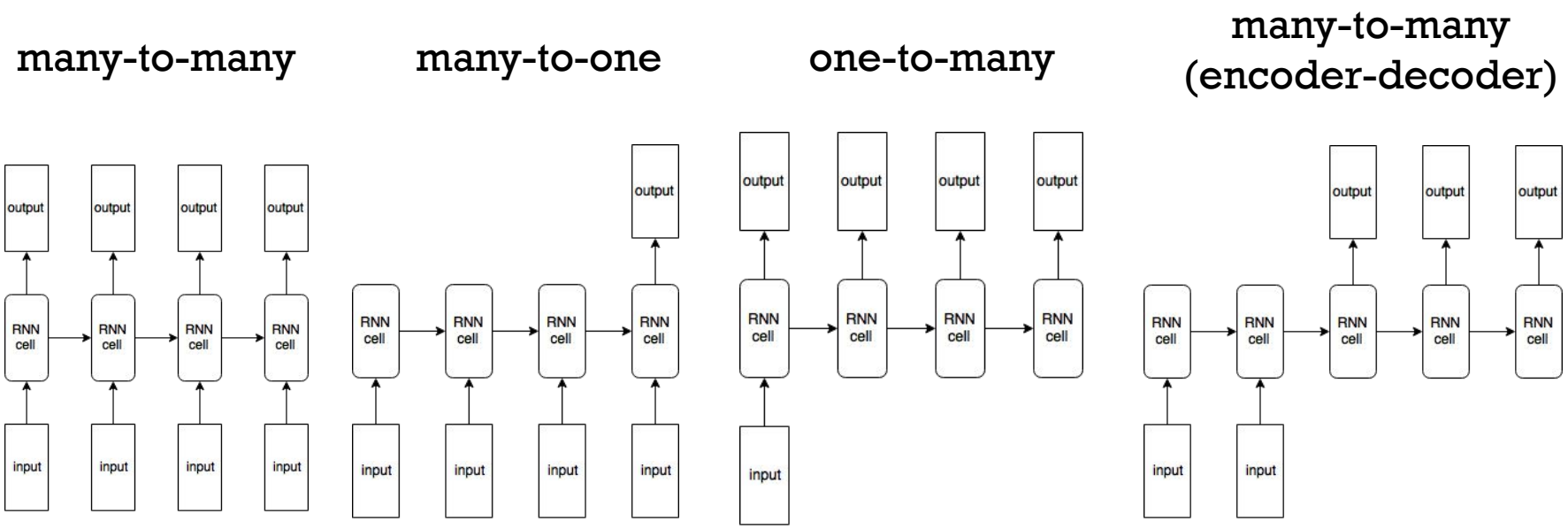
Classifier-based methods

- Have a pair of correction and features
  - whether/weather
  - Features:
    1. cloudy in +- 5 words
    2. Followed by “to VERB”
    3. Followed by “or not”

- End-to-end model using neural networks
  - Seq2Seq model (more on this in later lectures)

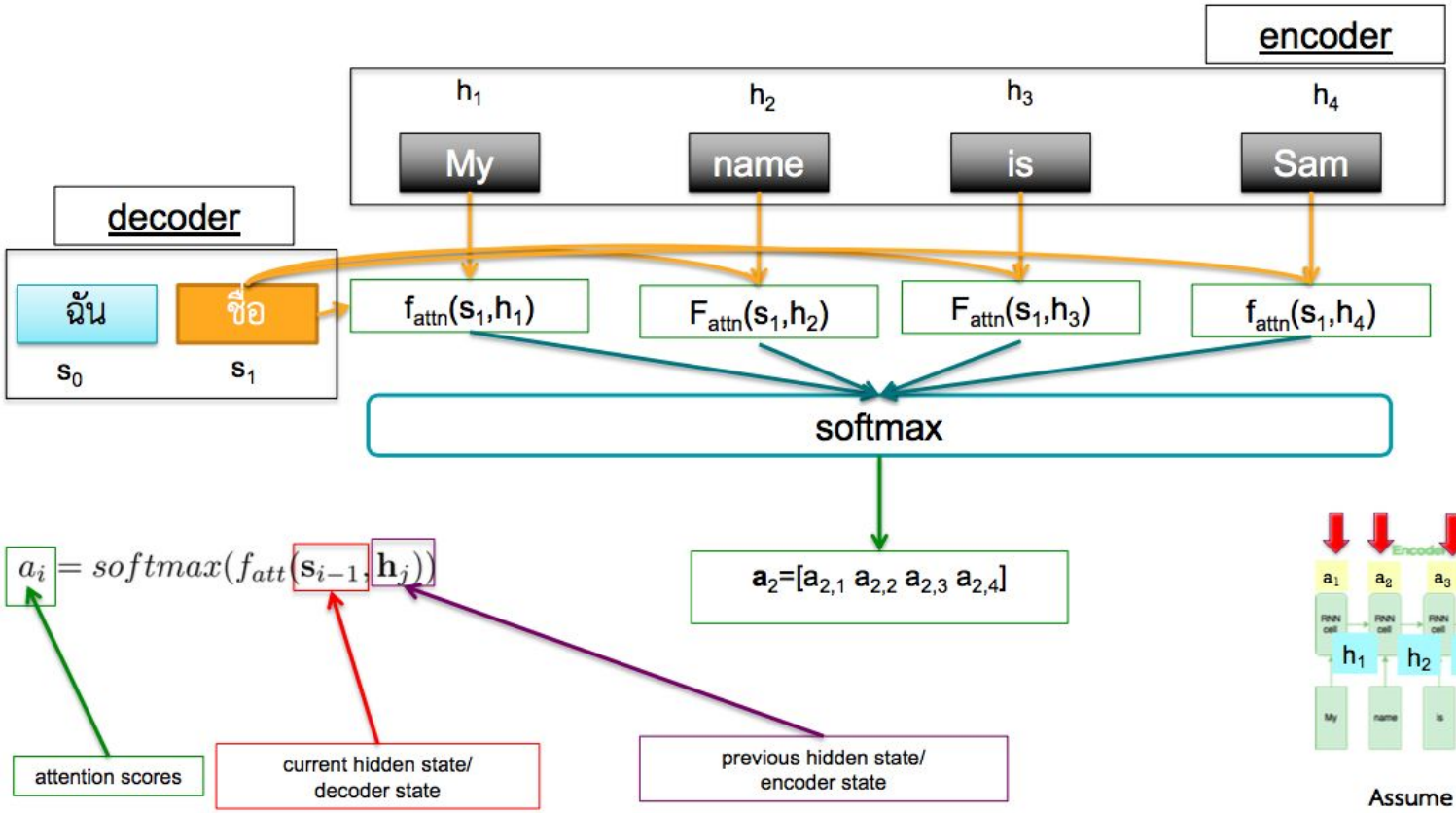
Review of attention and encoder-decoder

Different types of RNN architectures

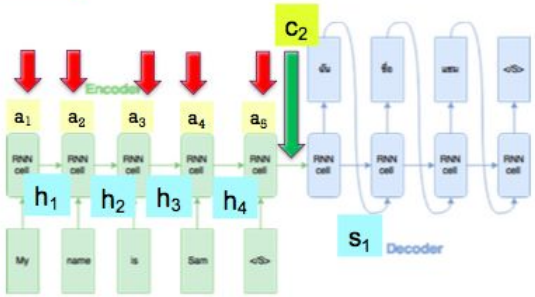


Reference: The Unreasonable Effectiveness of Recurrent Neural Networks, http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Attention Calculation Example (1): Attention Scores



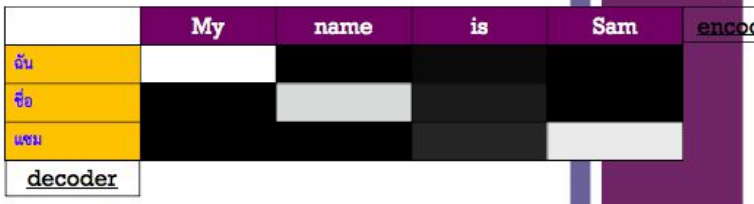
$$a_i = softmax(f_{att}(\mathbf{s}_{i-1}, \mathbf{h}_j))$$



Assume now try to translate "ชื่อ"
to give high weight at "name"

$$a_i = softmax(f_{att}(\mathbf{s}_{i-1}, \mathbf{h}_j))$$

Type of Attention mechanisms



(Remember that there are many variants of attention function f_{attn})

Additive attention: The original attention mechanism (Bahdanau et al., 2015) uses a one-hidden layer feed-forward network to calculate the attention alignment:

$$f_{attn}(\mathbf{s}_{i-1}, \mathbf{h}_j) = tanh(\mathbf{W}_a[\mathbf{s}_{i-1}; \mathbf{h}_j])$$

Multiplicative attention: Multiplicative attention (Luong et al., 2015) simplifies the attention operation by calculating the following function:

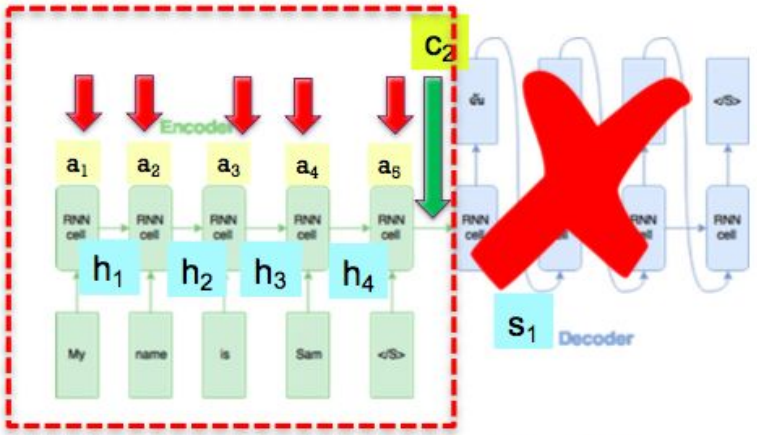
$$f_{attn}(\mathbf{s}_{i-1}, \mathbf{h}_j) = \mathbf{s}_{i-1}^{\top} \mathbf{W}_a \mathbf{h}_j$$

Self-attention: Without any additional information, however, we can still extract relevant aspects from the sentence by allowing it to attend to itself using self-attention (Lin et al., 2017)

$$\mathbf{a} = softmax(\mathbf{w}_{s_2} tanh(\mathbf{W}_{s_1} \mathbf{H}^T))$$

Key-value attention: key-value attention (Daniluk et al., 2017) is a recent attention variant that separates form from function by keeping separate vectors for the attention calculation.

Self attention

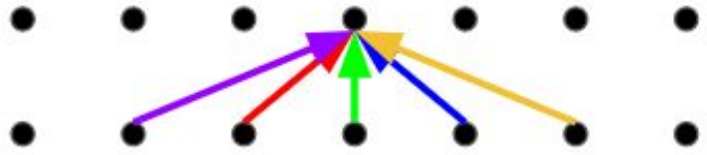


Assume now try to translate "ชื่อ"

to give high weight at "name"

No need for additional information in order to select where to attend

Convolution



Self-Attention

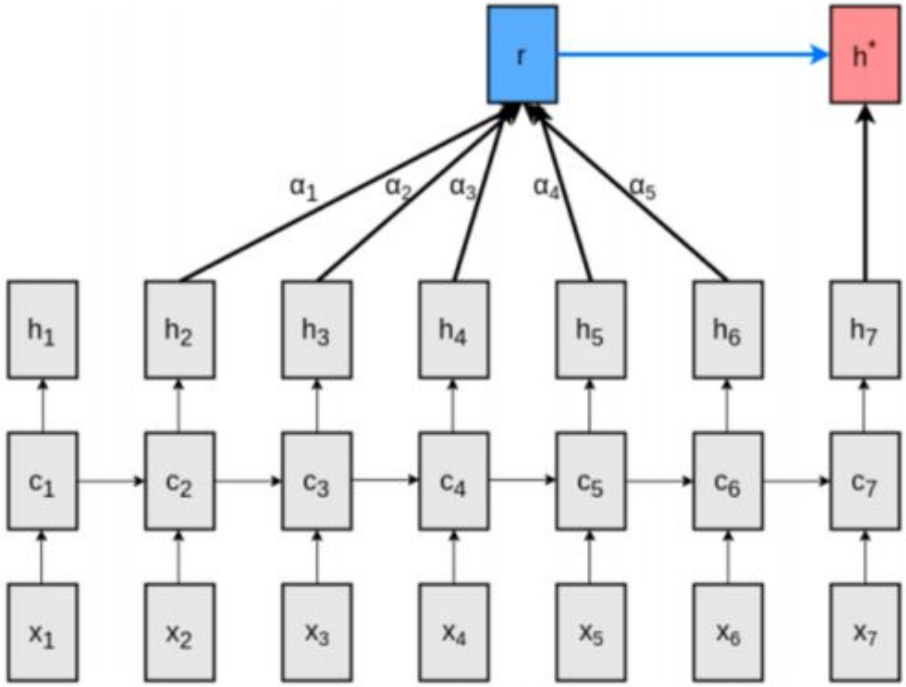
Similar to CNN in flow

https://nlp.stanford.edu/seminar/details/lkaiser.pdf

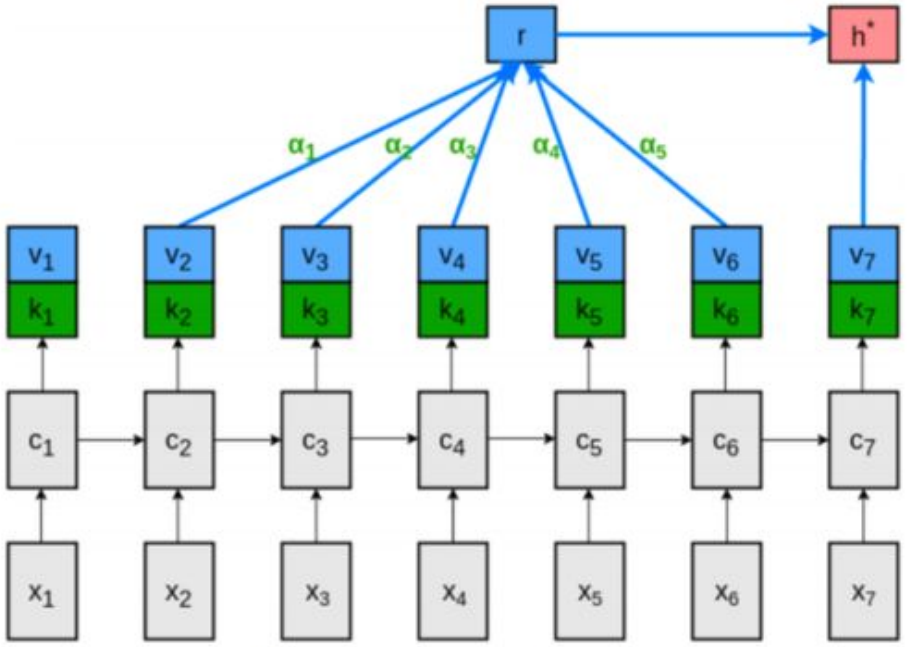
Key-value attention

Normal attention use the same vector to find the position and use as values

Use key to find the position, and use the values as information

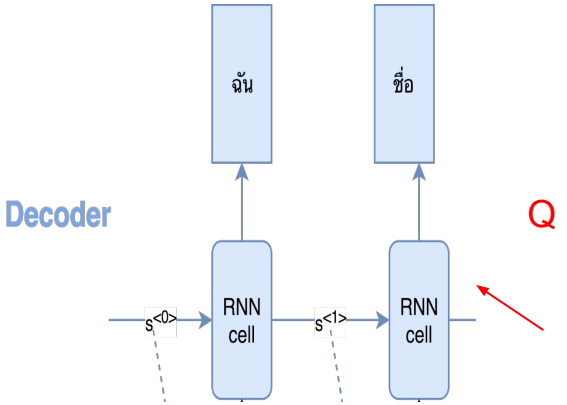


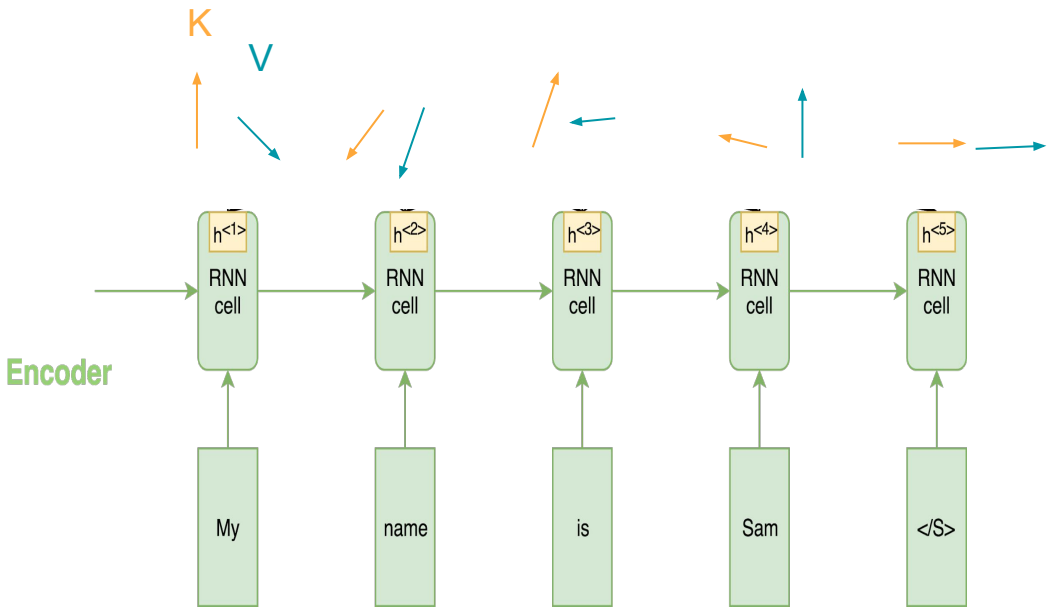
(a) Neural language model with attention.



(b) Key-value separation.

Pictorial view of KV attention





Type of errors in Thai social

Table 1: Examples of different types of errors and their respective correction.

Type of Error	Occurrence (%)	Error	Correction
Misspelling	61.38	ทุ๊กคน	ทุกคน
		คว่ำบัตร	คว่ำบาตร
Morphed	24.00	ครัช	ครับ
		ตั๊ลล๊าคคคค	น่ารัก
Abbreviation	15.00	มค	ม.ค.
		พน	พรุ่งนี้
Spoonerism	0.14	พับกบ	พบกับ
Slangs	0.08	ตีเนียน	No correction
		อ้อย	ทอดสะพาน
Other	5.63	โรบินสัน	No correction

Thai Spelling Correction and Word Normalization on Social Text Using a Two-Stage Pipeline With Neural Contextual Attention

Error detector

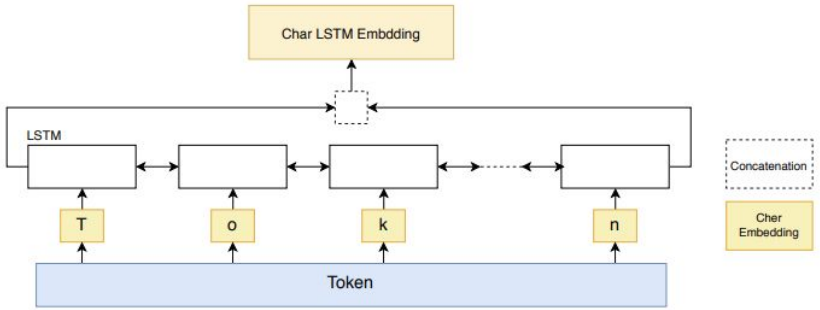


Figure 3: Character LSTM embedding layer encoding a word token.

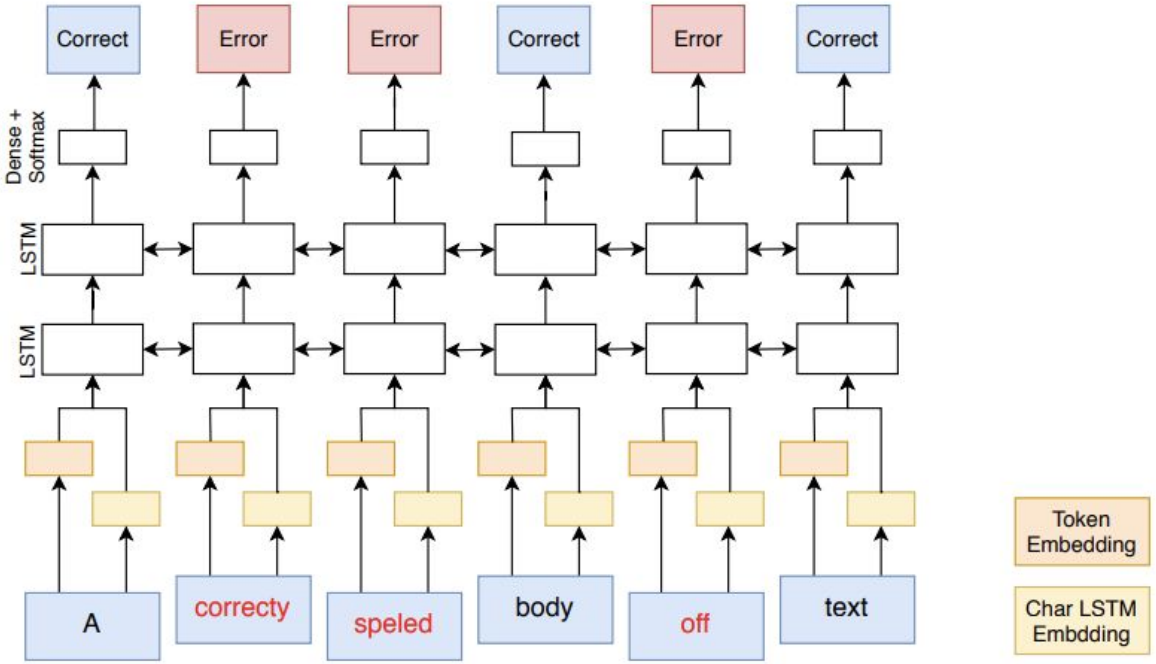
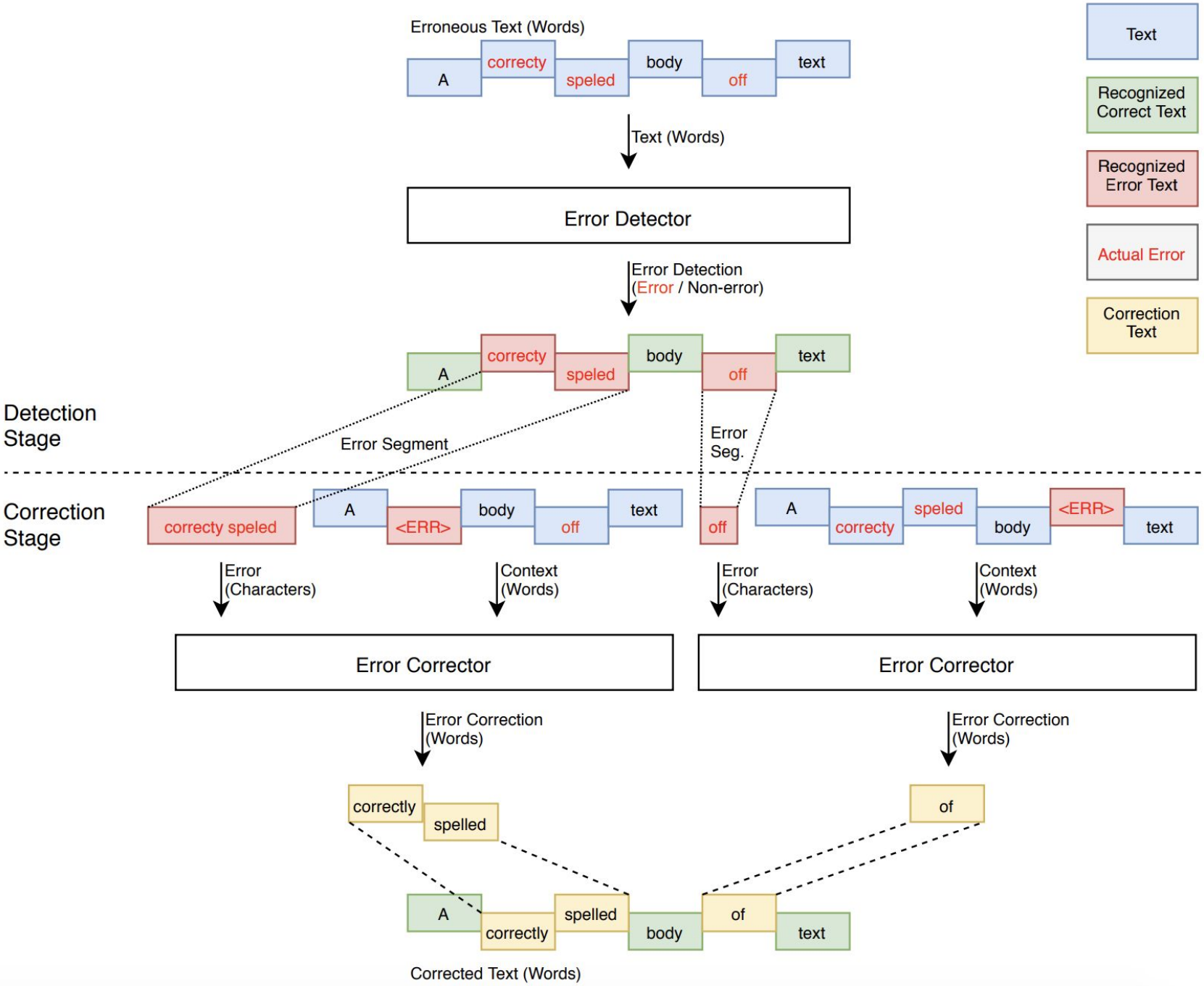


Figure 2: Error Detector operating on a sequence.


Erroneous Text (Words)
A
correcty
speled
body
off
text
Text (Words)
Error Detector
Error Detection
(Error / Non-error)
Detection
Stage
Error Segment
Error
Seg.
Correction
Stage
correcty speled
<ERR>
Error
(Characters)
Context
(Words)
Error Corrector
Error Correction
(Words)
correctly
spelled
of
Corrected Text (Words)
Text
Recognized
Correct Text
Recognized
Error Text
Actual Error
Correction
Text

Corrector

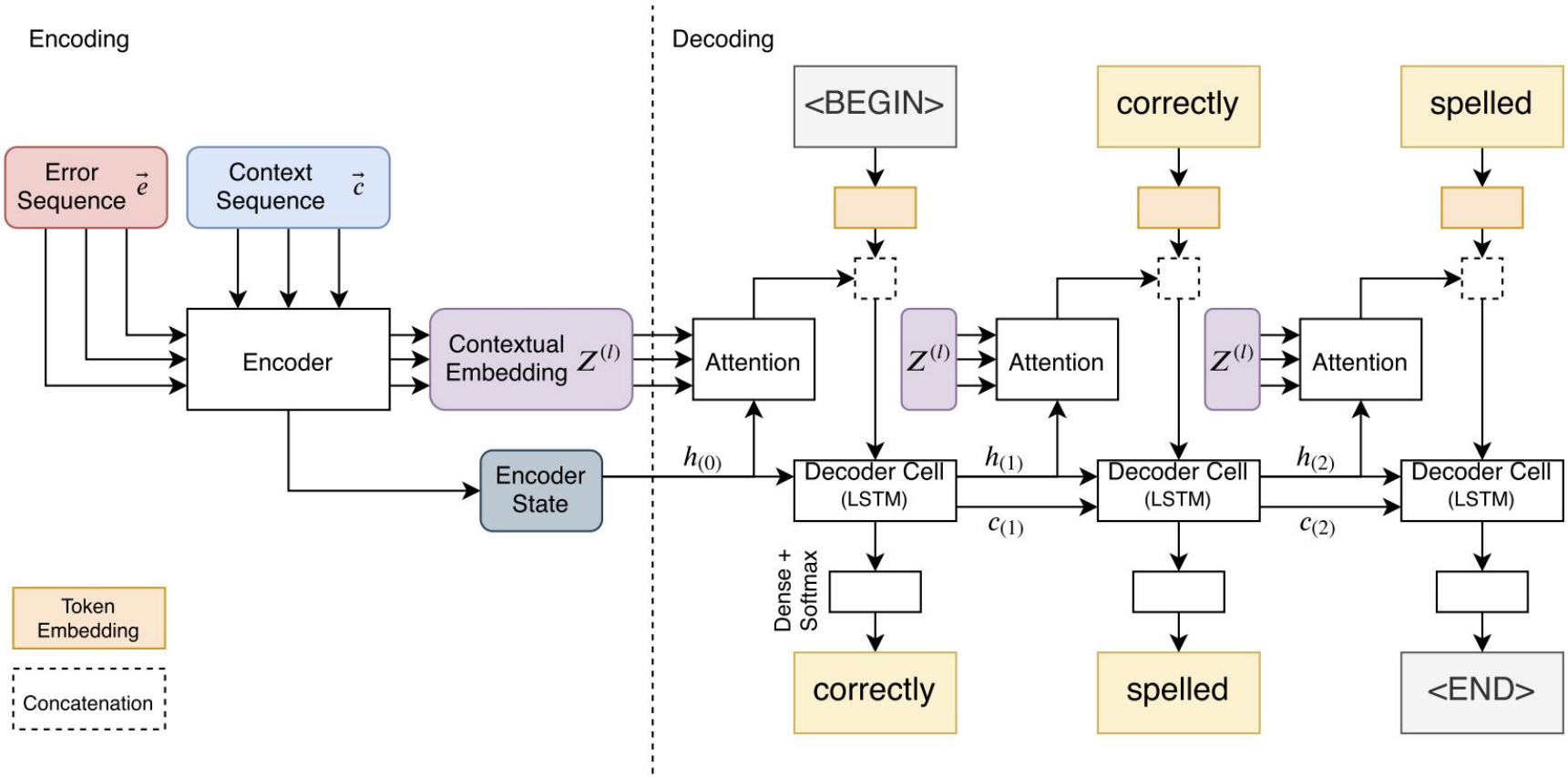
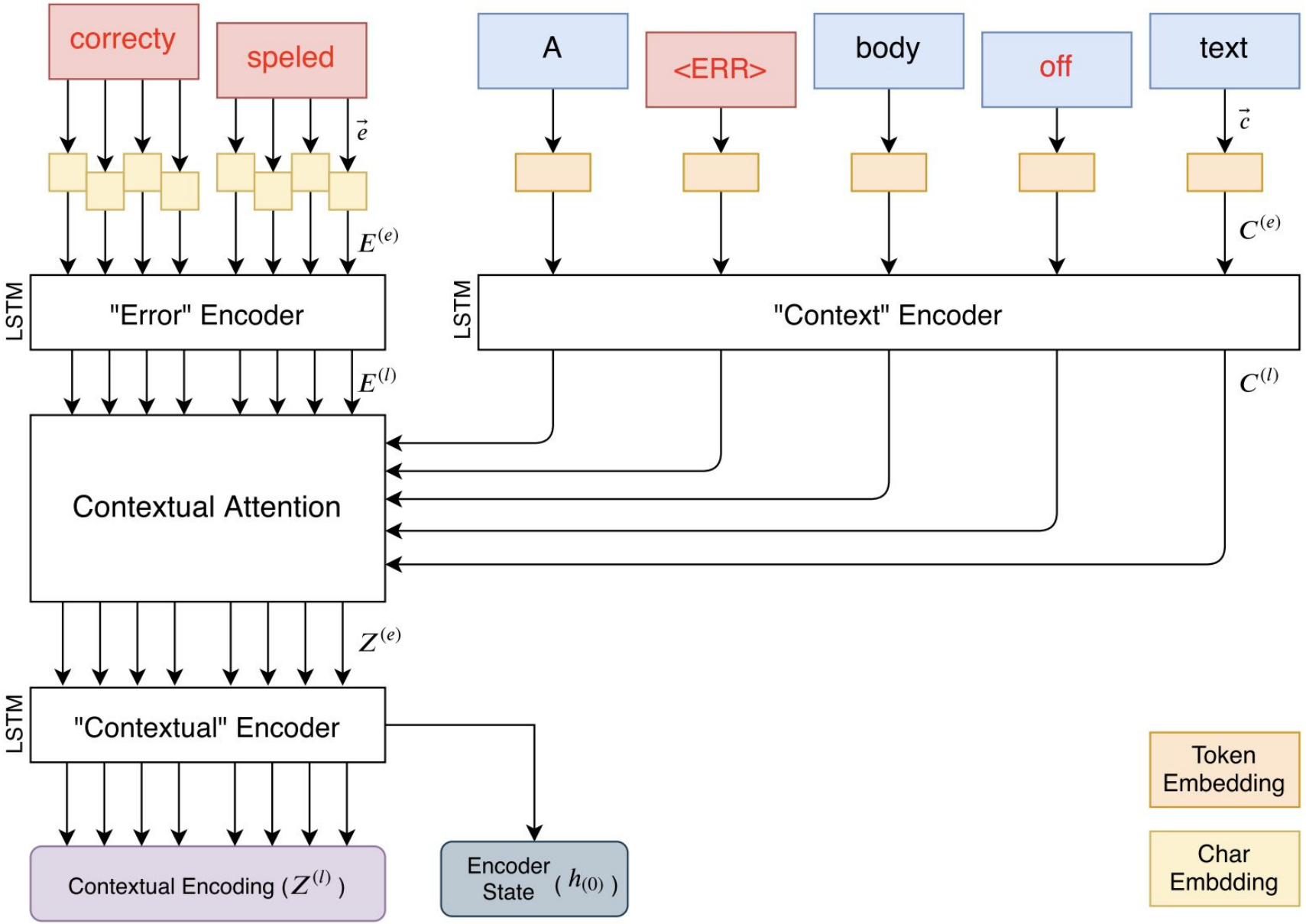


FIGURE 4. **Structure of the Seq2Seq text corrector. The model is split into two parts: the encoder and decoder, separated by the dotted line. This figure highlights the decoder; a more detailed view of the encoder is shown in Fig 5 . Three parallel arrows represent passing a sequence of vectors (a matrix), while a single arrow represents passing a single vector.**


correcty
speled
$\vec{e}$
A
<ERR>
body
off
text
$\vec{c}$
$E^{(e)}$
$C^{(e)}$
LSTM
"Error" Encoder
LSTM
"Context" Encoder
$E^{(l)}$
$C^{(l)}$
Contextual Attention
$Z^{(e)}$
LSTM
"Contextual" Encoder
Contextual Encoding ($Z^{(l)}$)
Encoder State ($h_{(0)}$)
Token Embedding
Char Embdding

Handling OOV

Need some way to handle oov tokens

Intput
I went Don Muang airport
I went <oov> <oov> airport

Output
I went to <oov> <oov> Airport

Unlike MT, we need some way to copy words from the input.

Seq2seq with copy-augmented

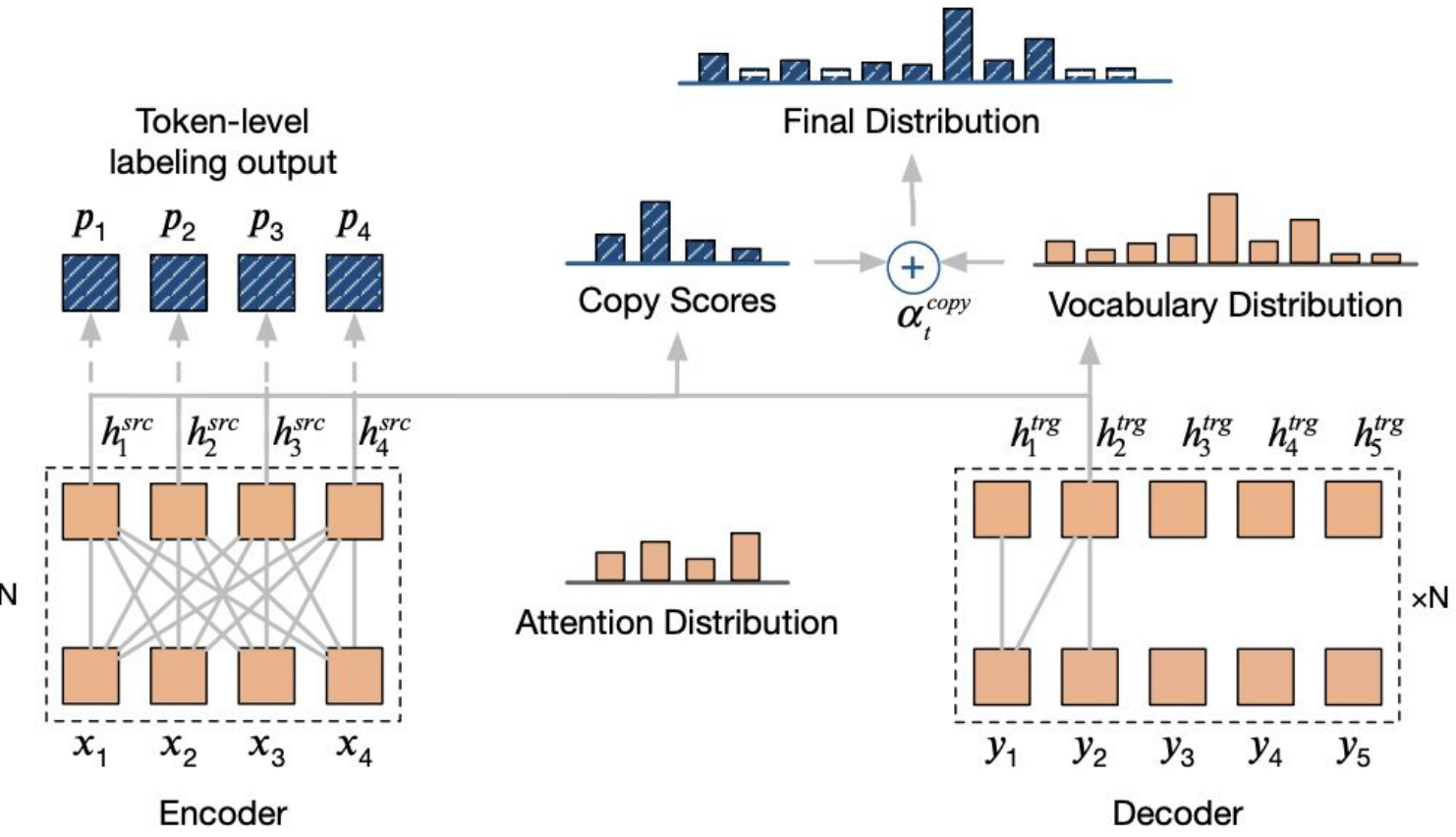


Figure 1: Copy-Augmented Architecture.

Also useful for text summarization

Improving Grammatical Error Correction via Pre-Training a Copy-Augmented Architecture with Unlabeled Data 2019

Incorporating Copying Mechanism in Sequence-to-Sequence Learning https://arxiv.org/abs/1603.06393

Table 3: Evaluation of end-to-end error correction on Thai UGWC test-set on various systems

Model	Type	GLEU	WER (%)	ΔWER (%)
Do nothing (source text)	-	0.8845	3.77	0.00
Ideal correction (Oracle)	-	1.0000	0.00	-100.00
Off the shelf				
Hunspell	2-Stage	0.8267	8.11	+115.12
PyThaiNLP	2-Stage	0.8612	5.58	+48.01
Trained				
Hunspell	2-Stage	0.8598	5.57	+47.75
Bi-GRU (180 token limit)	E2E	0.4035	50.82	+1,247.92
Bi-GRU (20 token limit)	E2E	0.7404	15.97	+323.57
Bi-GRU (50 token limit)	E2E	0.7462	17.51	+364.38
Copy-Aug Transformer*	E2E	0.9374	2.58	-31.56
Copy-Aug Transformer**	E2E	0.9409	2.51	-33.42
Ours	2-Stage	0.9453	2.24	-40.66
Ours*	2-Stage	0.9361	2.83	-25.03
Ours**	2-Stage	**0.9502**	**2.07**	**-45.21**
Ours** (with Oracle Detection)	2-Stage	0.9774	1.08	-71.39

* Model is only trained on noise injected dataset.
** Model is pre-trained on noise injected dataset before fine-tuned on the regular training-set.

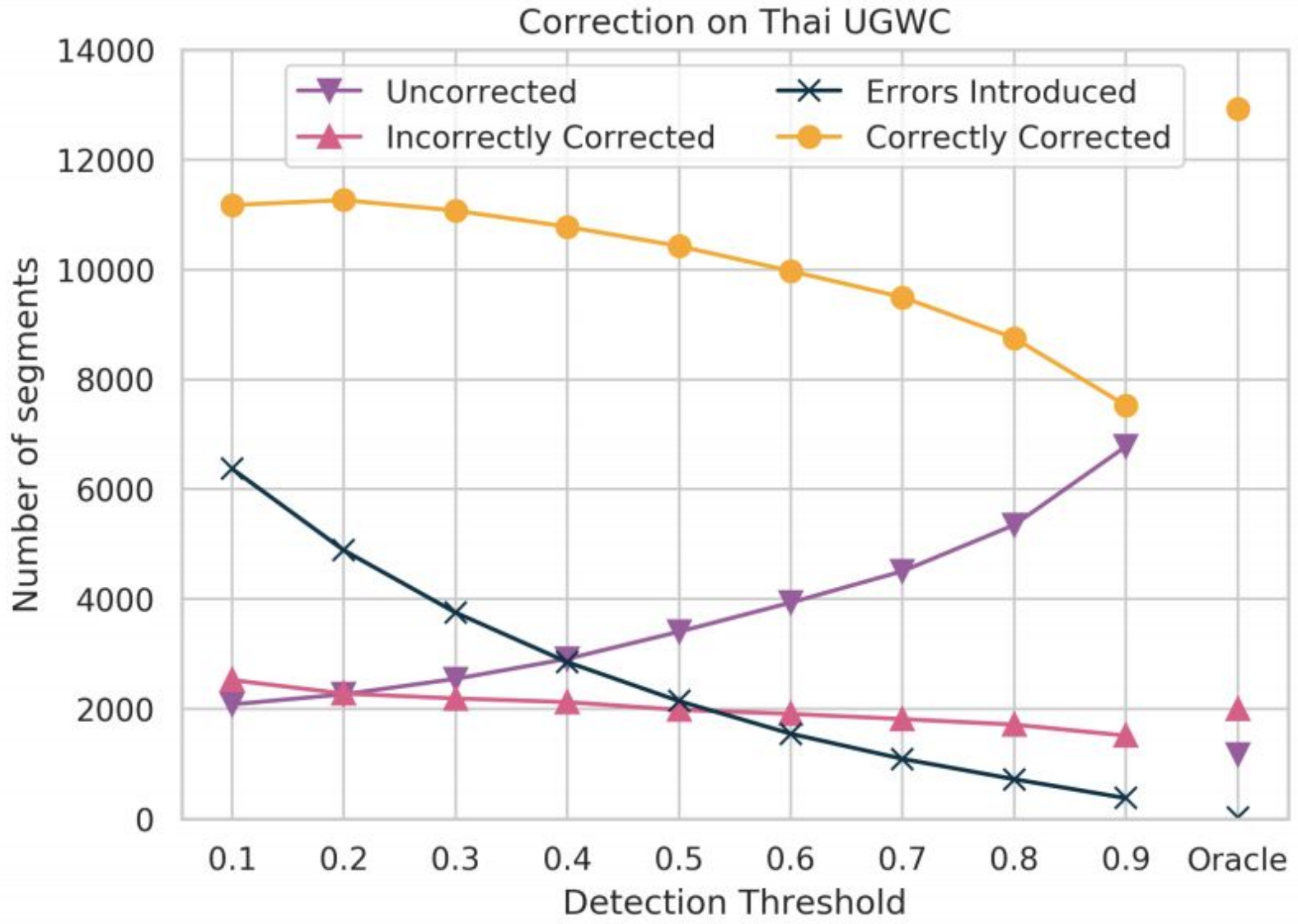


Figure 14: Types of corrections produced from our method on the Thai UGWC test-set

Issues with tokenization

Table 16 Correction error analysis of misspelling on the TC task on the Thai UGWC dataset

	Error Types	Issues with correction	Erroneous text	Labels	Correction produced
	Misspelling				
1	Single-word	Corrected as non-existent word	ล๊อค	ล็อก	ล็อค
2		Contextually incorrect (real word error)	อยาง	อยาก	อย่าง
3			สตาง	สตางค์	ส|.|ต่าง
4			ยอก|เลิด	ยกเลิก	ยอด
5			ไช่	ใช่	ใช้
6		Uncorrected	กาแลคซี่	กาแล็กซี	<OOV>
7	Single-word + incorrect tokenization		ฅนล้า	คนล้า	<OOV>
8			นู๋พยายาม	หนูพยายาม	<OOV>
9		Dropping word(s)	ค่พอไป|ตีดต่อ	ค่ะ|พอ|ไป|ติดต่อ	พอไป|ติดต่อ
10	Single-word + punctuation + incorrect tokenization		สมัค|sm|s	สมัคร| |sms	สมัคร
11	Multi-word	Partially correct	ยุ|ตรุรกี	อยู่|ตุร|กี	อยู่|กรณี
12			ได้|หรอ	ได้|หรือ	ได้|หรอ

Misspellings can caused mis-tokenization which cause bad corrections.

Providing groundtruth segmentations can alleviate this problem.

This is on-going research.

Misspelling invariant tokenizations?

If misspelling is a huge pain can we deal with it?

Make embeddings misspelling invariant

Idea: embedding of misspellings should be close to correct word

Misspelling Oblivious Word Embeddings https://www.aclweb.org/anthology/N19-1326/

Negative sampling review

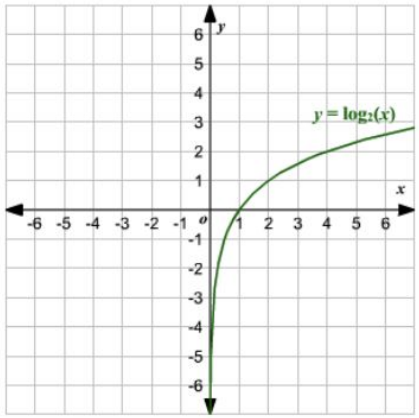
2) Skip-gram (cont.)
Negative Sampling (Binary Classification)

- The objective function for skip-gram with negative sampling:

$$J_t(\theta) = \log \sigma\left(u_o^T v_c\right) + \sum_{i=1}^{k} \mathbb{E}_{j \sim P(w)} \left[\log \sigma\left(-u_j^T v_c\right)\right]$$

Context word (positive, +1)

Negative samples (negative, -1)



σ = sigmoid

Misspelling oblivious loss

$$L_{W2V} := \sum_{i=1}^{|T|} \sum_{w_c \in C_i} [\ell(s(w_i, w_c)) + \sum_{w_n \in N_{i,c}} \ell(-s(w_i, w_n))]$$

skip-gram with negative samples

$$L_{SC} := \sum_{(w_m, w_e) \in M} [\ell(\hat{s}(w_m, w_e)) + \sum_{w_n \in N_{m,e}} \ell(-\hat{s}(w_m, w_n))]$$

misspelt is close to correct word than negative

$$L_{MOE} := (1 - \alpha) L_{FT} + \alpha \frac{|T|}{|M|} L_{SC}$$

combine the two losses

Transformer

Transformer and others

Overview

- Applications
  - Grammar correction
  - QA
- Techniques
  - Transformer
  - Beam search
  - GAN for text

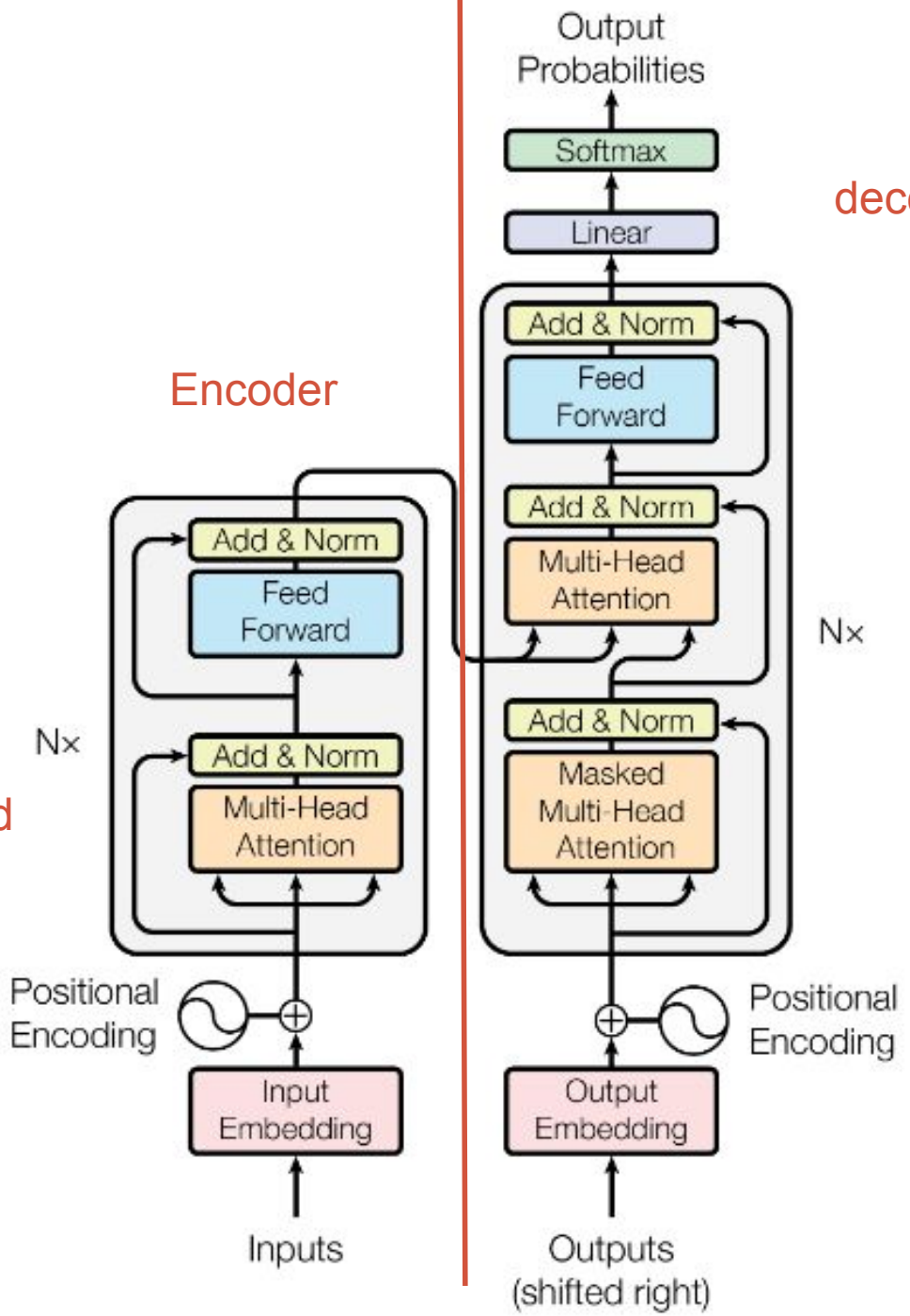
Attention is all you need

Abstract

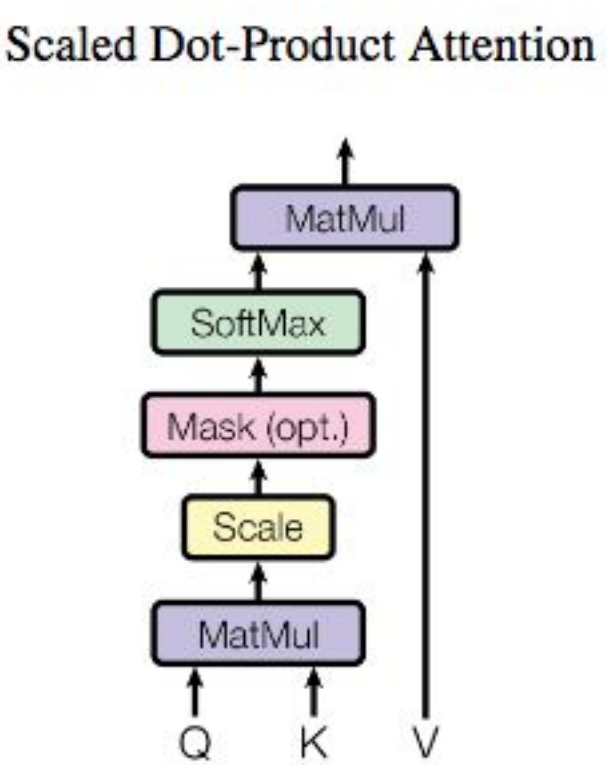
The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, 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.

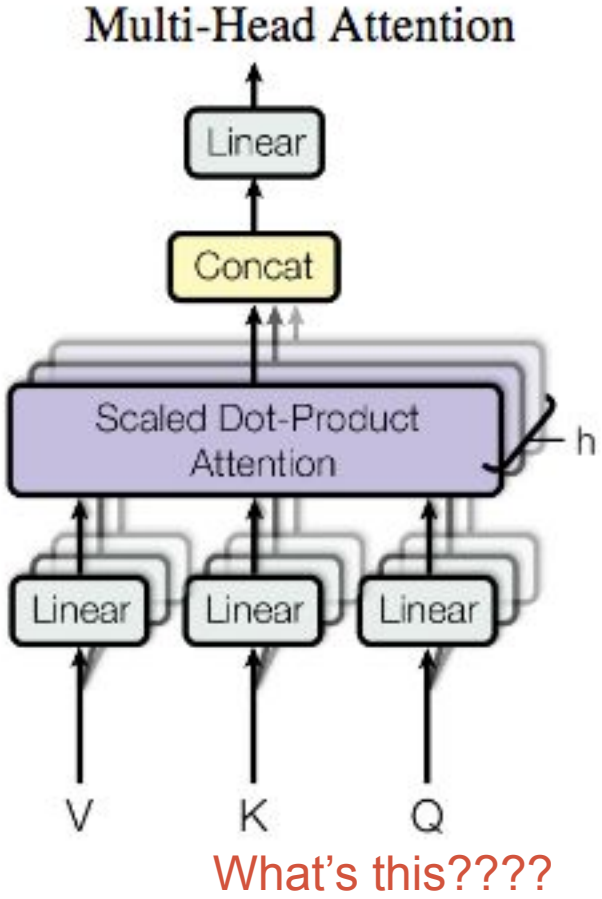
https://arxiv.org/pdf/1706.03762.pdf

To eliminate GRU which remembers time, encode position instead



Multi-head attention





What's this????

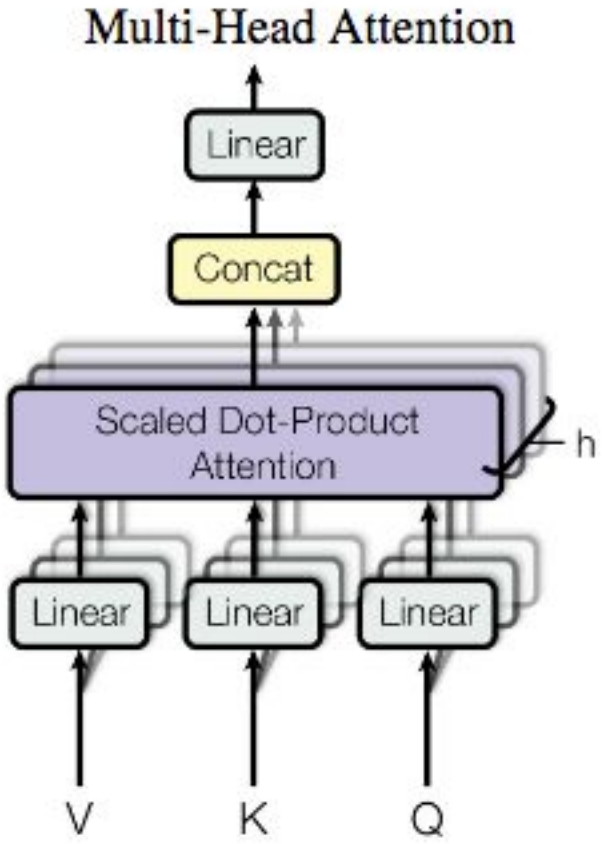
Query – used with Key to determine the position
Value – used as the information after determining the position

Attention drawback

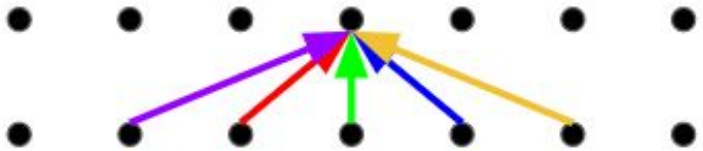
- Convolution: weights * input. Each weights are different. So position is encoded.
- Self-attention: a weighted average. Position information is lost at the output

Multi-head attention

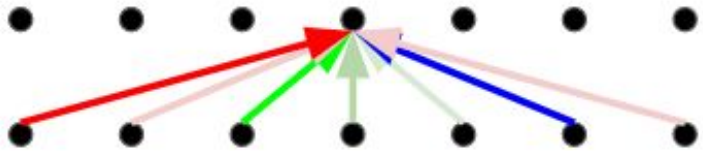
- Multiple attention layers (heads) that run in parallel
- Each head use different weights
- Each head can learn different relationship



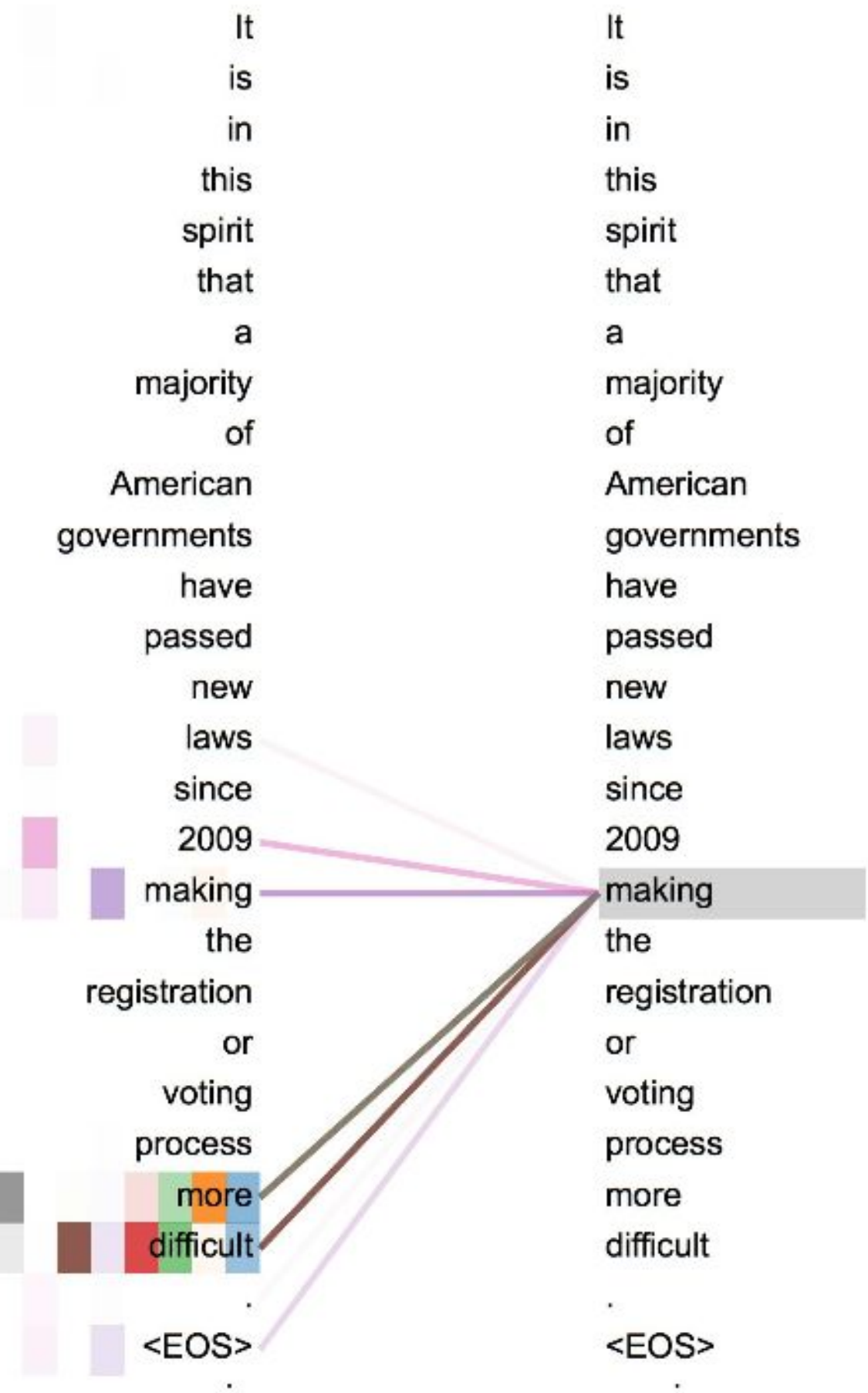
Convolution

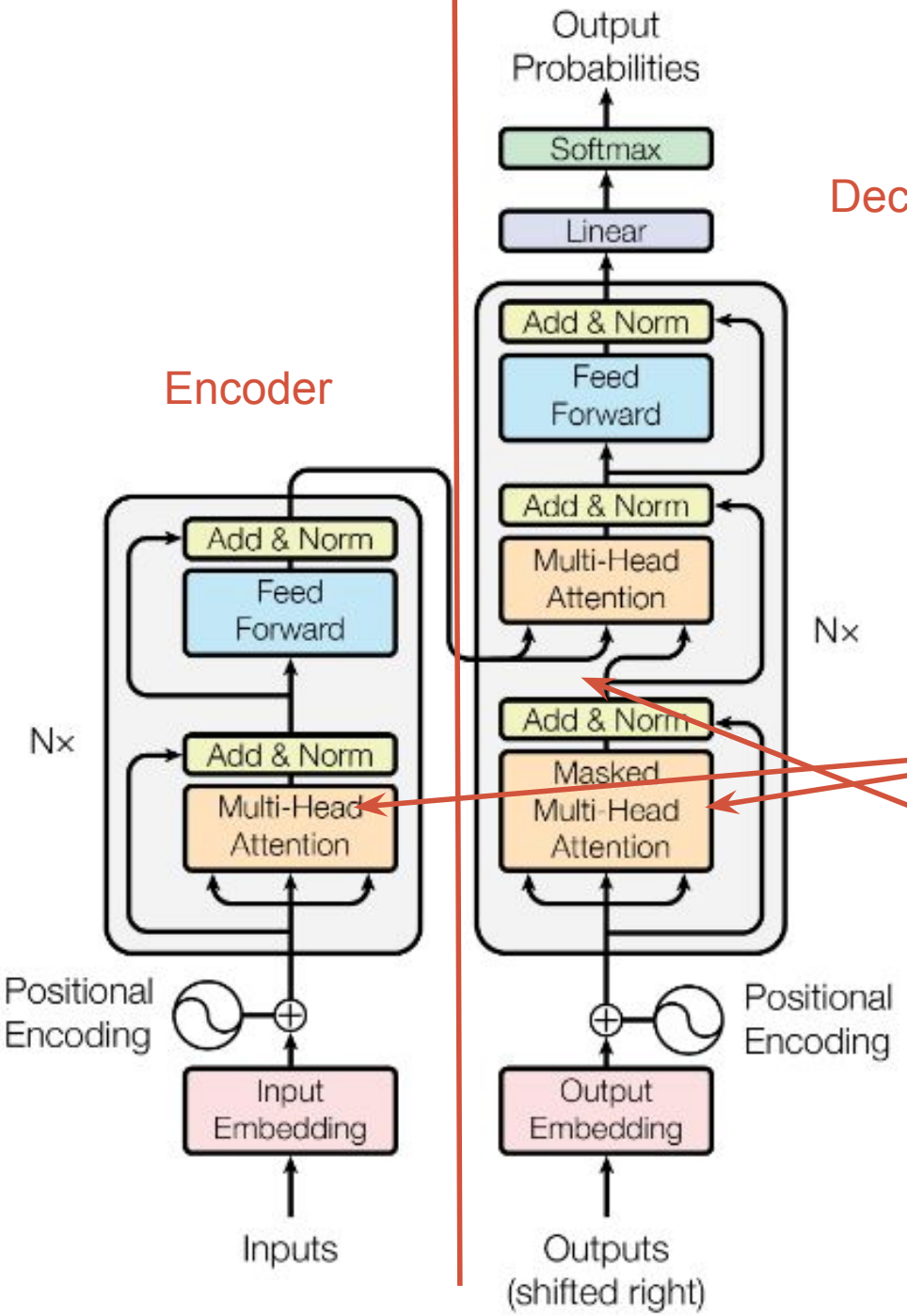


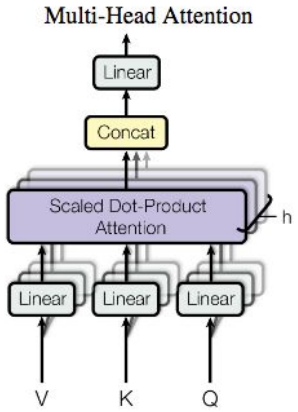
Multi-Head Attention



Multi-head visualization

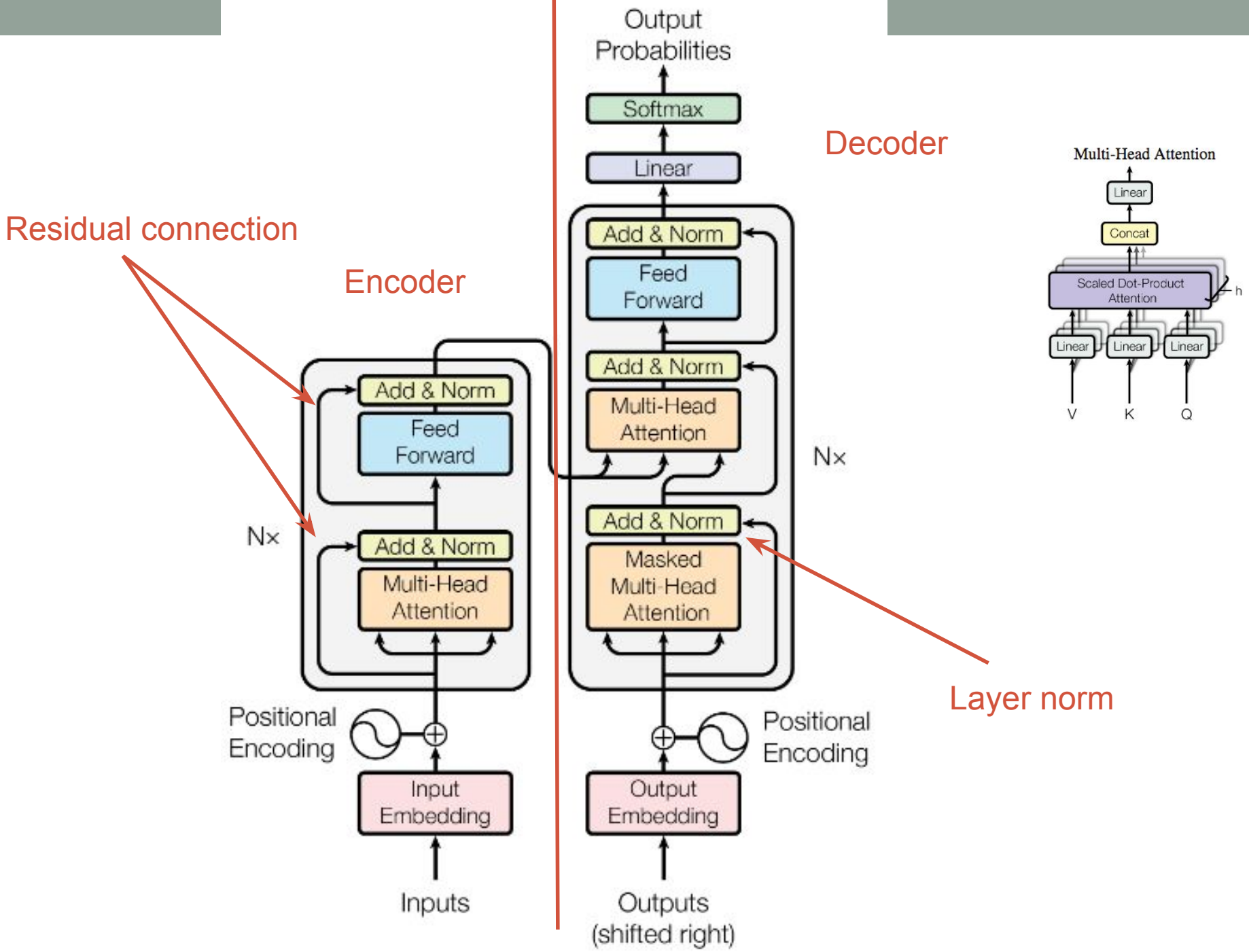






Self attention

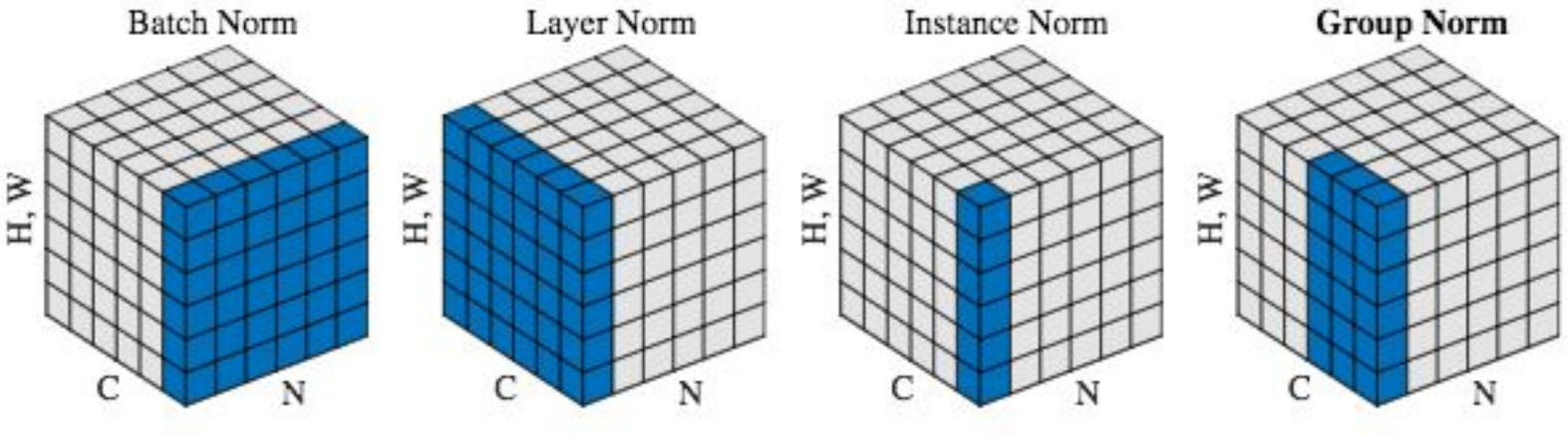
Encoder gives Value and Key
The decoder gives Query


Residual connection
Encoder
Decoder
Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)
Layer norm
Multi-Head Attention
Linear
Concat
Scaled Dot-Product Attention
h
Linear
Linear
Linear
V
K
Q

Layer norm

- Normalize the mean and SD
- Batch norm vs layer norm vs Instance norm vs group norm

Group is used to distributed models into multiple GPUs

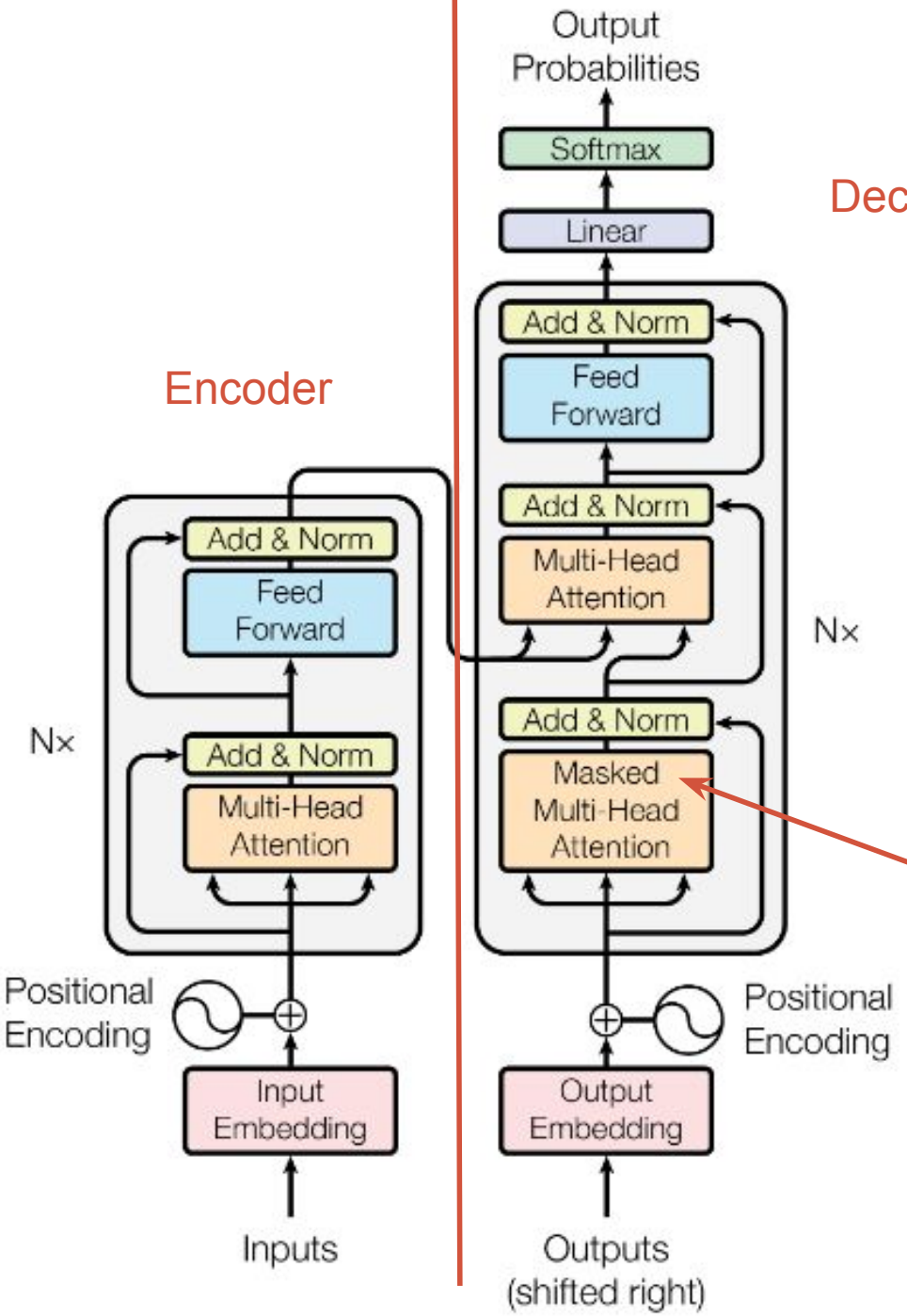


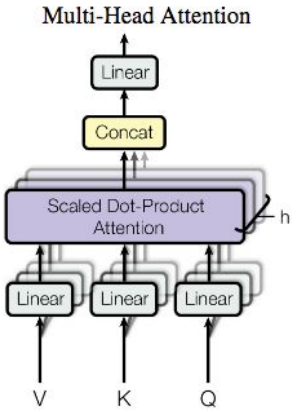
N – example in mini batch
C – Channel output
H,W – spatial coordinates (x,y)

Box is output tensor from CNN

BN and GN are usually best, GN is better when batch size is small (Vision task)

https://arxiv.org/abs/1803.08494





Prevent looking ahead (cheating)

MT results

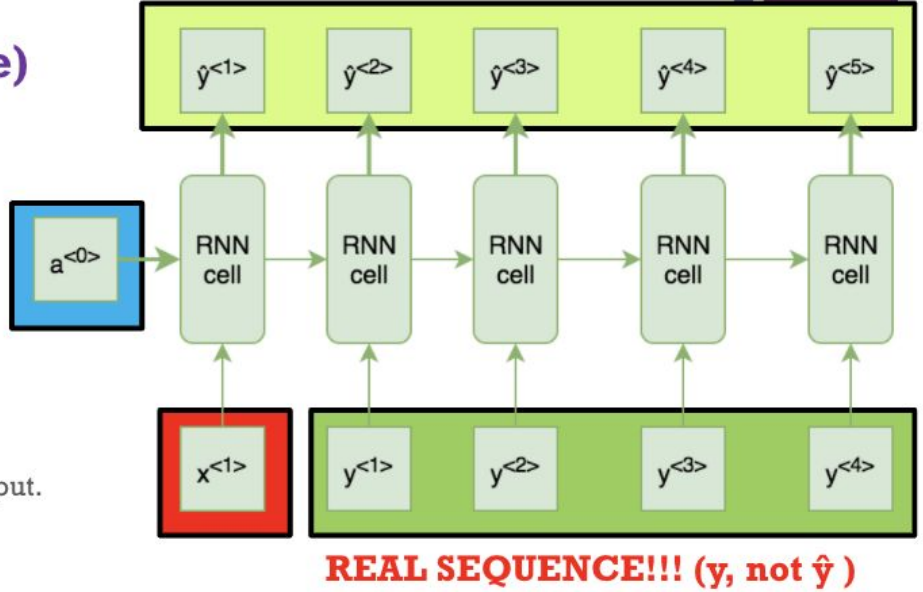
Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	**$3.3 \cdot 10^{18}$**	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

Can use for other tasks, like ASR, parsing, etc.

Text generation model (training)

- **One-to-Many RNN (autoregressive)**
  - The only real input is $x^{<1>}$
  - $a^{<0>}$ is the initial hidden state.
  - $\hat{y}$ is the predicted output.
  - y is an actual output.
  - During **the training phase**, instead of using the predicted output to feed into the next time-step, we use the actual output.

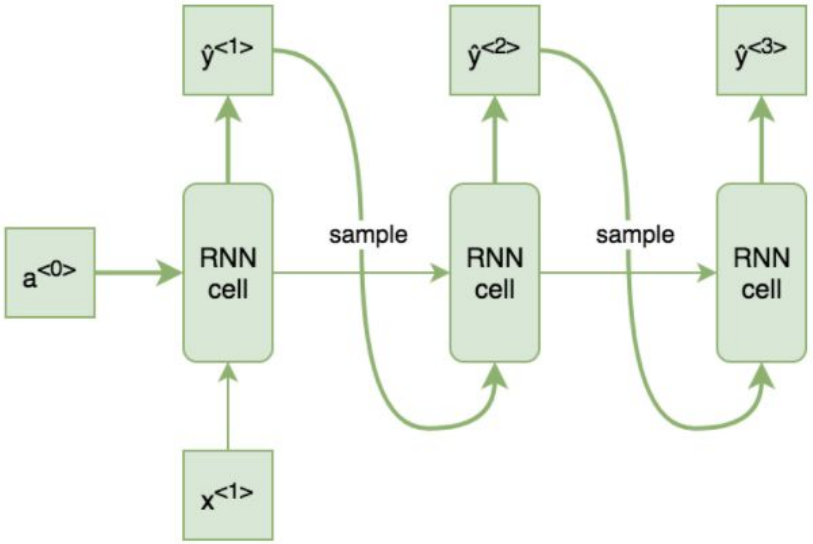


$$a^{<t>} = Wa^{<t-1>} + Wx^{<t>} + b$$

Text generation model (inference)

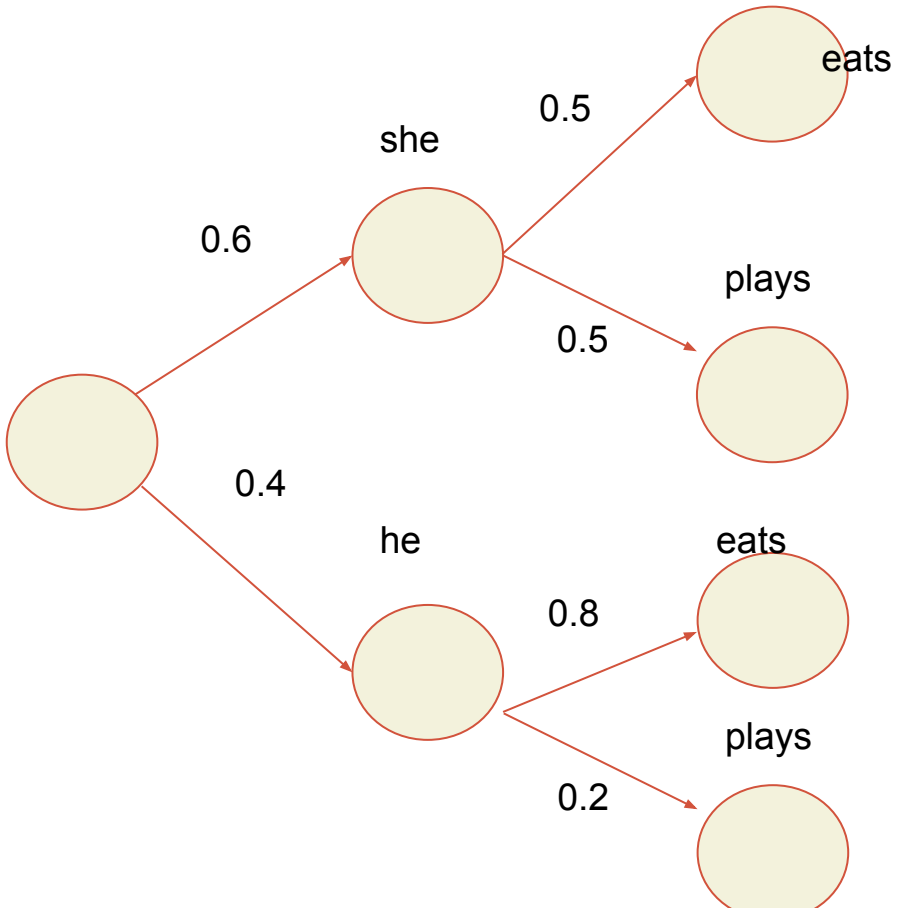
Training | Inference

- To generate a novel sequence, the inference model (testing phase) randomly samples an output from a softmax distribution.



Beam search

Autoregressive requires making a decision even though it might not be optimal for the whole sequence



However, expanding the tree will be exponential

P(he eats) = 0.32
P(she eats) = 0.30

Beam search

Beam search keeps a list of n options at all times

Problems

Longer sentences have lower probabilities.

- Normalize by length
- See homework

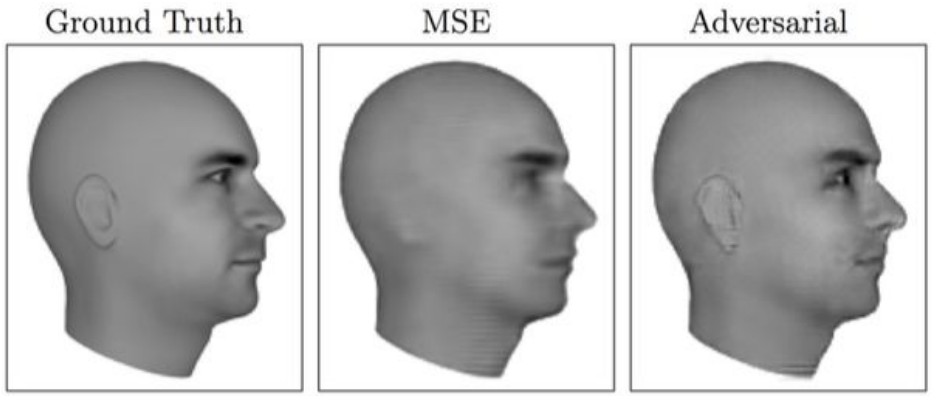
Generative Adversarial Networks (GANs)

Learning distributions

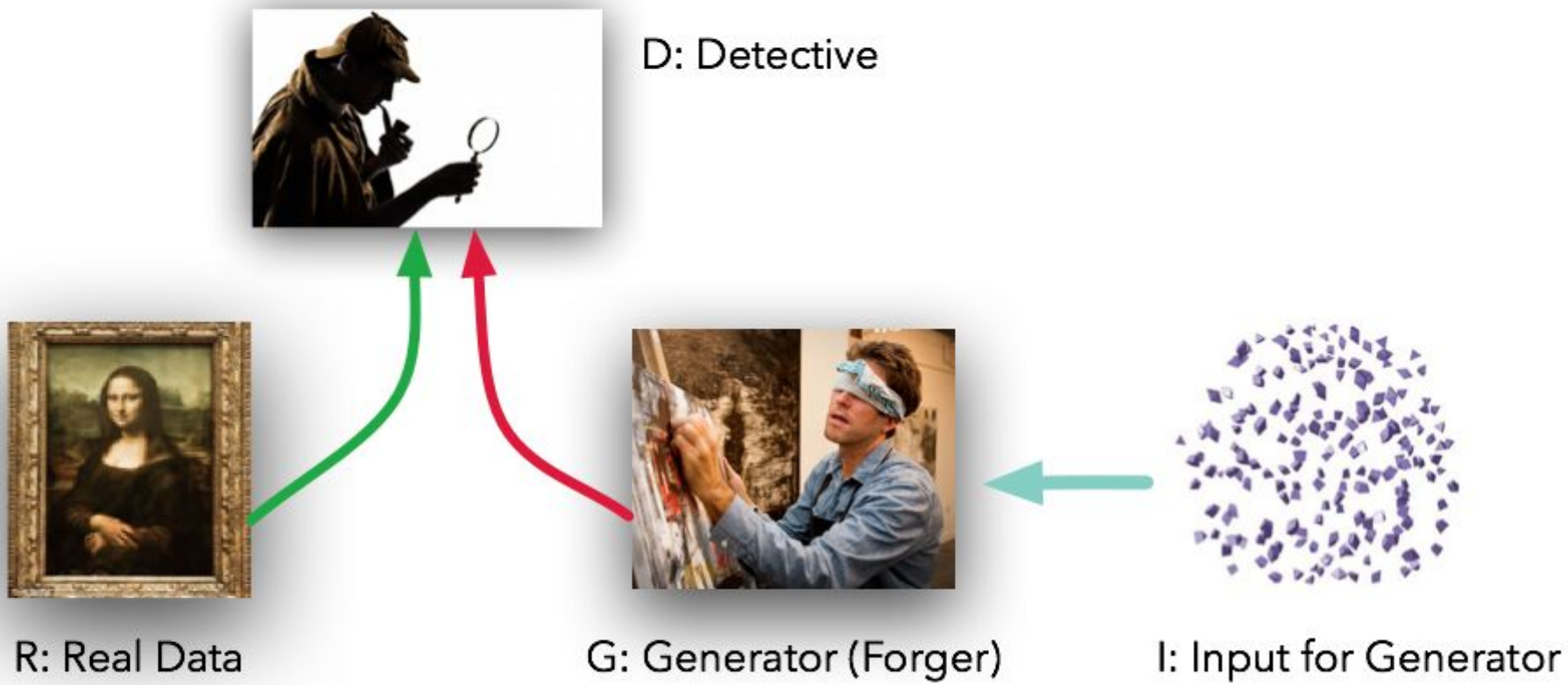
- Supervised learning tasks usually have one correct answer

Learning distributions

- Supervised learning tasks usually have one correct answer
- Sometimes there are more than one possibility
  - What is the next frame of a video?
  - What is the missing pixels in an image?
  - What word is missing from the blank?
    - I eat ____

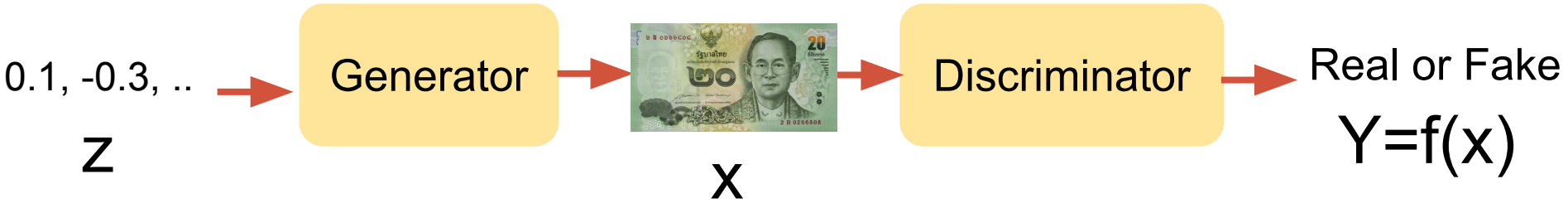


https://arxiv.org/abs/1701.00160



https://medium.com/@devnag/generative-adversarial-networks-gans-in-50-lines-of-code-pytorch-e81b79659e3f

Generative Adversarial Networks (GANs)



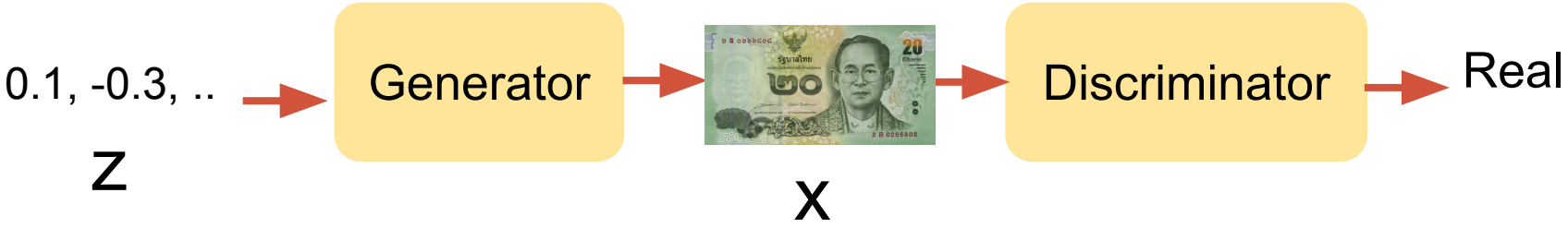
Consider a money counterfeiter

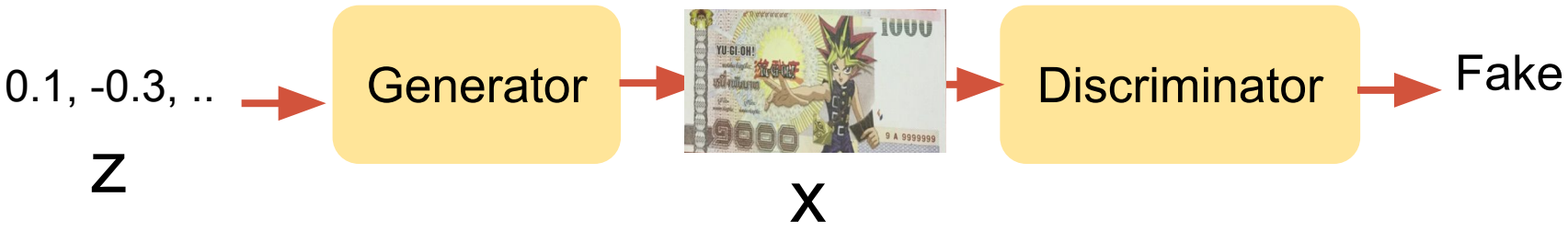
He wants to make fake money that looks real

There's a police that tries to differentiate fake and real money.

The counterfeiter is the adversary and is generating fake inputs. – Generator network

The police is try to discriminate between fake and real inputs. – Discriminator network


0.1, -0.3, ..
z
Generator
x
Discriminator
Real


0.1, -0.3, ..
z
Generator
x
Discriminator
Fake

real x

Discriminator

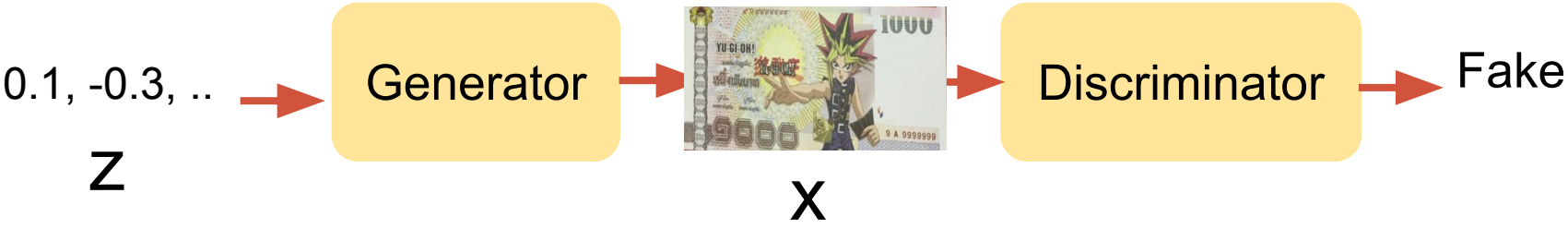
Real

gradient

fake x

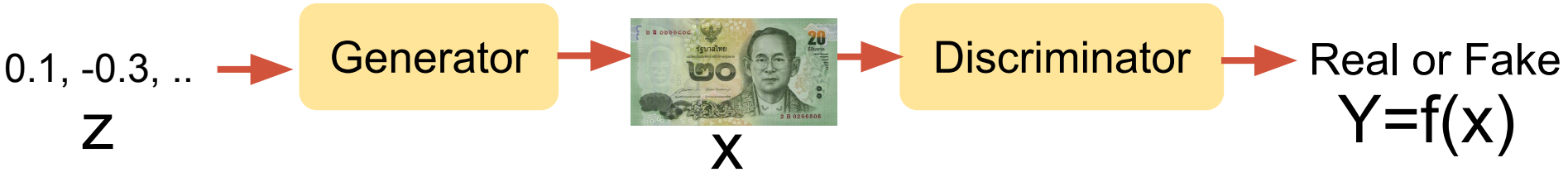
The discriminator learns to differentiate real and fake data

Learns by backpropagation



The generator learns to be better by the gradient given by the discriminator

Generative Adversarial Networks (GAN)



- Generator (Money Faker):
  - Maximize Y

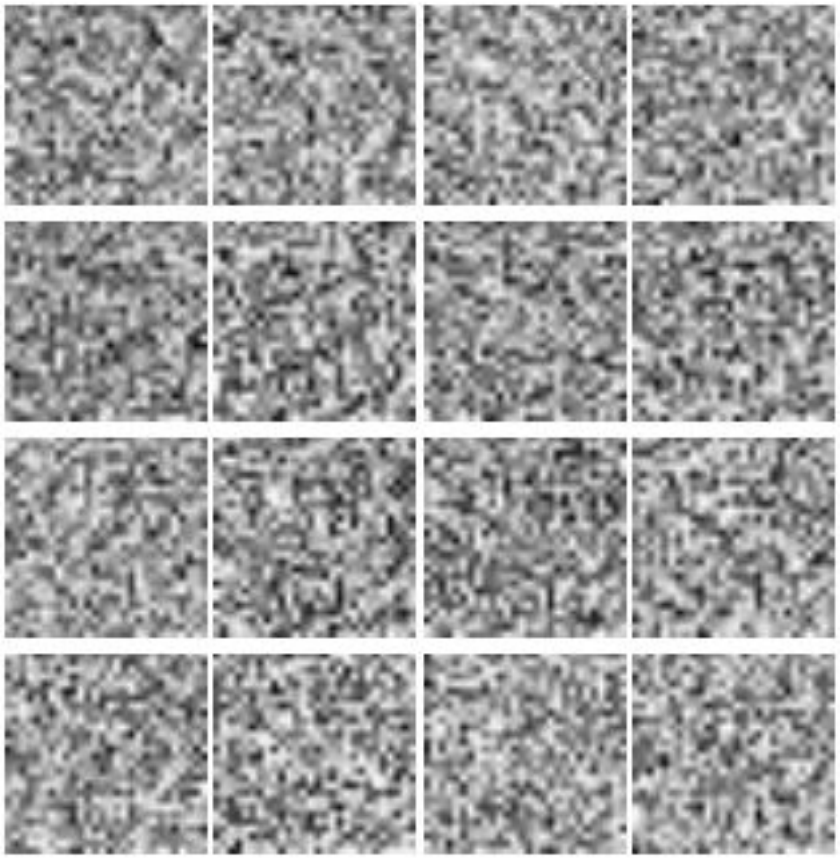
$$\min_G \mathbb{E}_{\mathbf{z}}[\log(1 - D(G(\mathbf{z}))]$$

- Discriminator (Police):
  - For real images => Maximize Y
  - For generated images from the faker => Minimize Y

$$\max_D \mathbb{E}_{\mathbf{z}}[\log(1 - D(G(\mathbf{z}))] + \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}}[\log(D(\mathbf{x}))]$$

GAN example

Generator output starts from random noise and gets better as we train.



https://wiseodd.github.io/techblog/2016/09/17/gan-tensorflow/

GANs Loss Formulations

$$\max_D \mathbb{E}_{\mathbf{z}}[\log(1 - D(G(\mathbf{z}))] + \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[\log(D(\mathbf{x}))]$$

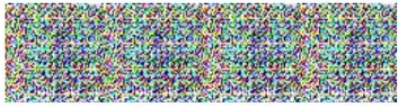
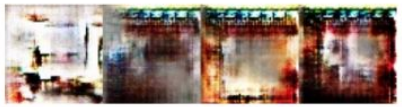
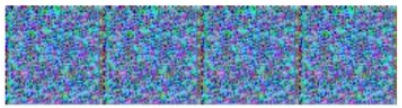
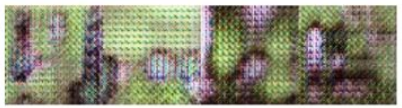
Discriminator

$$\min_G \mathbb{E}_{\mathbf{z}}[\log(1 - D(G(\mathbf{z}))]$$

Generator

GAN	Discriminator Loss	Generator Loss
MM GAN	$\mathcal{L}_{\text{D}}^{\text{GAN}} = -\mathbb{E}_{x\sim p_d}[\log(D(x))] - \mathbb{E}_{\hat{x}\sim p_g}[\log(1 - D(\hat{x}))]$	$\mathcal{L}_{\text{G}}^{\text{GAN}} = \mathbb{E}_{\hat{x}\sim p_g}[\log(1 - D(\hat{x}))]$
NS GAN	$\mathcal{L}_{\text{D}}^{\text{NSGAN}} = -\mathbb{E}_{x\sim p_d}[\log(D(x))] - \mathbb{E}_{\hat{x}\sim p_g}[\log(1 - D(\hat{x}))]$	$\mathcal{L}_{\text{G}}^{\text{NSGAN}} = -\mathbb{E}_{\hat{x}\sim p_g}[\log(D(\hat{x}))]$
WGAN	$\mathcal{L}_{\text{D}}^{\text{WGAN}} = -\mathbb{E}_{x\sim p_d}[D(x)] + \mathbb{E}_{\hat{x}\sim p_g}[D(\hat{x})]$	$\mathcal{L}_{\text{G}}^{\text{WGAN}} = -\mathbb{E}_{\hat{x}\sim p_g}[D(\hat{x})]$
WGAN GP	$\mathcal{L}_{\text{D}}^{\text{WGANGP}} = \mathcal{L}_{\text{D}}^{\text{WGAN}} + \lambda\mathbb{E}_{\hat{x}\sim p_g}[(\|\|\nabla D(\alpha x + (1-\alpha\hat{x})\|\|_2 - 1)^2]$	$\mathcal{L}_{\text{G}}^{\text{WGANGP}} = -\mathbb{E}_{\hat{x}\sim p_g}[D(\hat{x})]$
LS GAN	$\mathcal{L}_{\text{D}}^{\text{LSGAN}} = -\mathbb{E}_{x\sim p_d}[(D(x) - 1)^2] + \mathbb{E}_{\hat{x}\sim p_g}[D(\hat{x})^2]$	$\mathcal{L}_{\text{G}}^{\text{LSGAN}} = -\mathbb{E}_{\hat{x}\sim p_g}[(D(\hat{x} - 1))^2]$
DRAGAN	$\mathcal{L}_{\text{D}}^{\text{DRAGAN}} = \mathcal{L}_{\text{D}}^{\text{GAN}} + \lambda\mathbb{E}_{\hat{x}\sim p_d+\mathcal{N}(0,c)}[(\|\|\nabla D(\hat{x})\|\|_2 - 1)^2]$	$\mathcal{L}_{\text{G}}^{\text{DRAGAN}} = \mathbb{E}_{\hat{x}\sim p_g}[\log(1 - D(\hat{x}))]$
BEGAN	$\mathcal{L}_{\text{D}}^{\text{BEGAN}} = \mathbb{E}_{x\sim p_d}[\|\|x - \text{AE}(x)\|\|_1] - k_t\mathbb{E}_{\hat{x}\sim p_g}[\|\|\hat{x} - \text{AE}(\hat{x})\|\|_1]$	$\mathcal{L}_{\text{G}}^{\text{BEGAN}} = \mathbb{E}_{\hat{x}\sim p_g}[\|\|\hat{x} - \text{AE}(\hat{x})\|\|_1]$

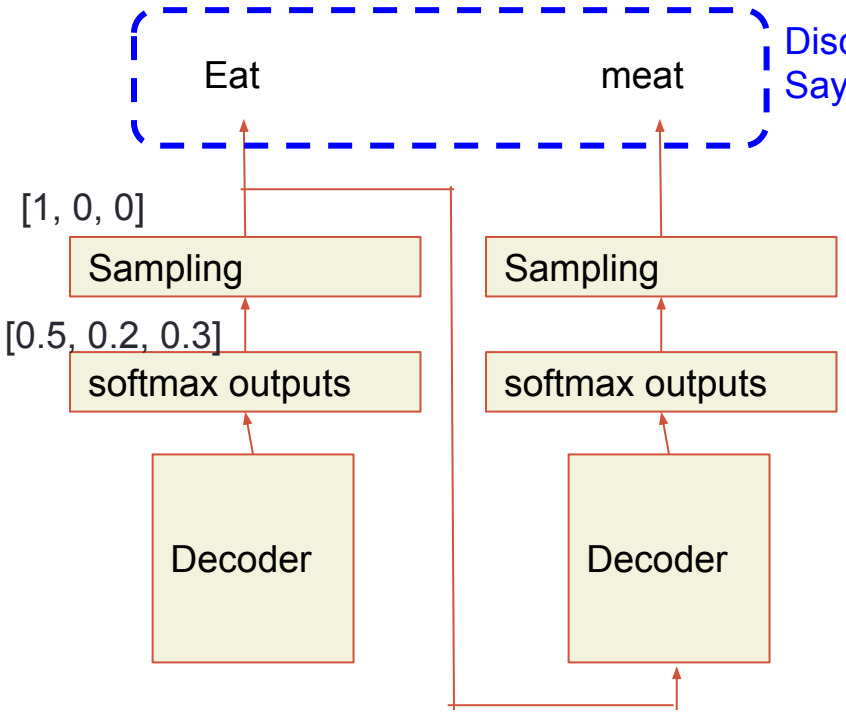
Are GANs Created Equal? A Large-Scale Study [Lucic et al. 2018]

DCGAN	LSGAN	WGAN (clipping)	WGAN-GP (ours)
Baseline (G: DCGAN, D: DCGAN)			
G: No BN and a constant number of filters, D: DCGAN				
G: 4-layer 512-dim ReLU MLP, D: DCGAN				
No normalization in either G or D				
Gated multiplicative nonlinearities everywhere in G and D				
tanh nonlinearities everywhere in G and D				
101-layer ResNet G and D				

GAN for text generation

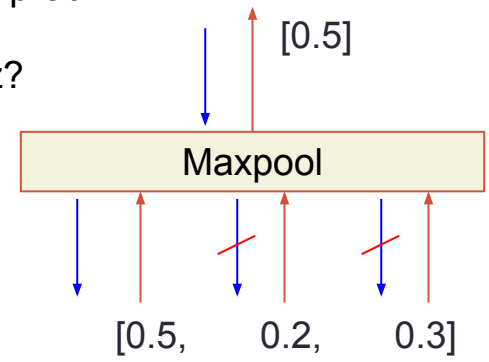
Autoregressive decoding includes a sampling process

Cannot gradient descent



softmax prob [0.3 0.4 0.1 0.2] -> word index z
Pick the max
Pick randomly based on prob

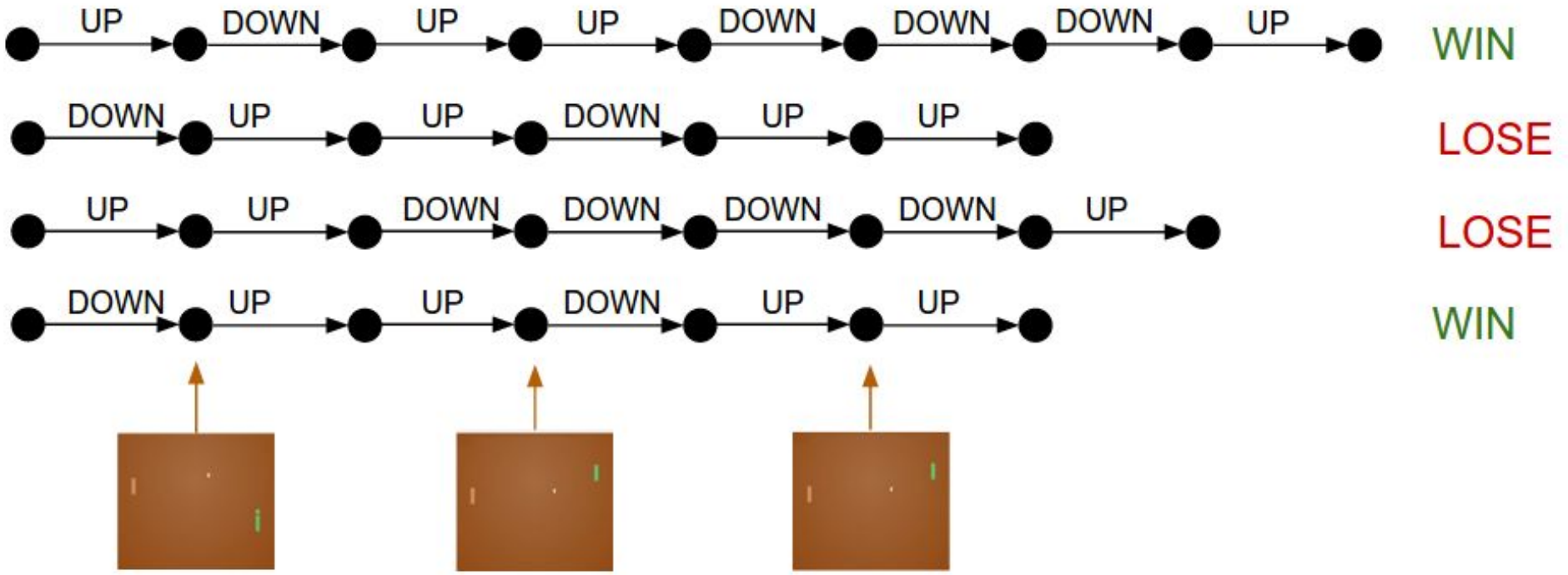
What is the gradient of z?



Two popular methods: REINFORCE, Gumbel-Softmax approximation (https://arxiv.org/abs/1611.01144)

RL and policy gradients

- Credit assignment problem in reinforcement learning



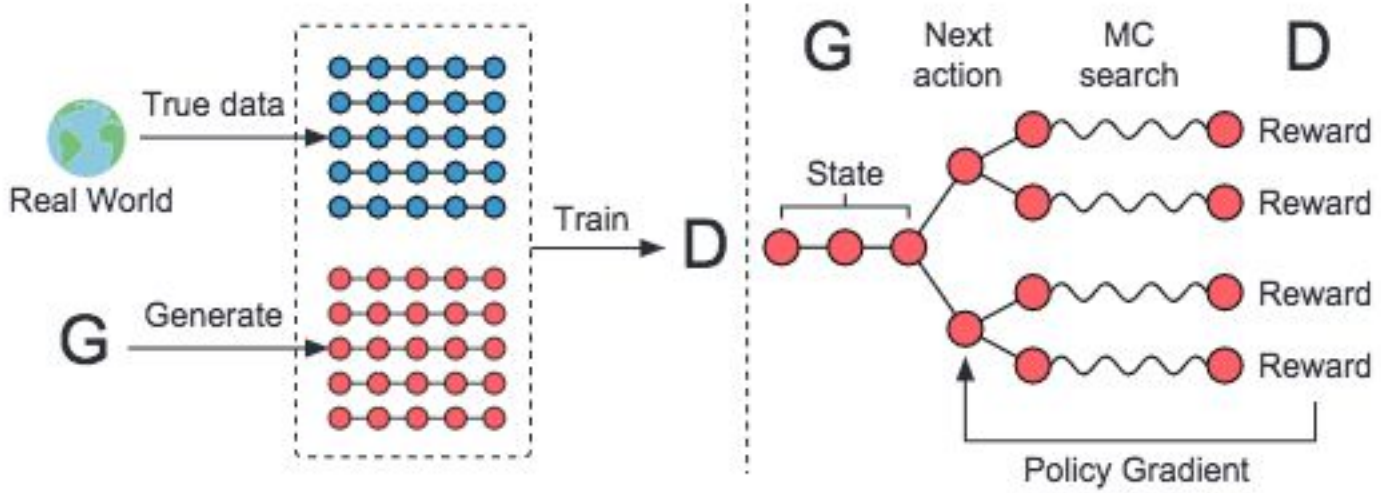
which move makes you win?

For RL with policy gradients, we increase the probability of every move that results in a win (REINFORCE algorithm)

http://karpathy.github.io/2016/05/31/rl/

GAN with text generation (SeqGAN)

- Use policy gradient to update the generator (the agent in RL setting)
- The discriminator (critic) gives the reward



How is this different from our previous text generation? (Maximum likelihood)
Want to generate exact vs Want to generate “real” sentences

https://arxiv.org/pdf/1609.05473.pdf

Table 2: Chinese poem generation performance comparison.

Algorithm	Human score	p-value	BLEU-2	p-value
MLE	0.4165	0.0034	0.6670	$< 10^{-6}$
SeqGAN	**0.5356**		**0.7389**	
Real data	0.6011		0.746	

Table 3: Obama political speech generation performance.

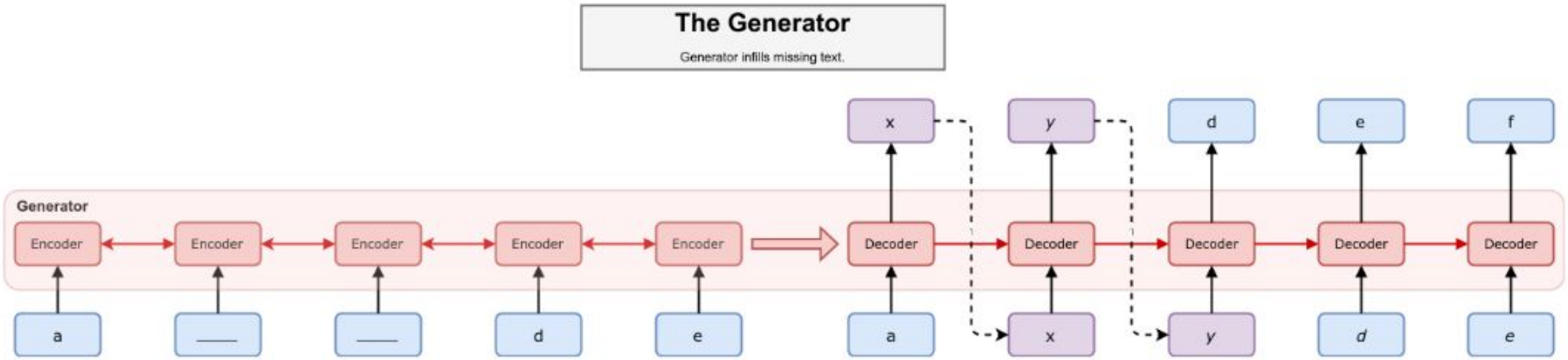
Algorithm	BLEU-3	p-value	BLEU-4	p-value
MLE	0.519	$< 10^{-6}$	0.416	0.00014
SeqGAN	**0.556**		**0.427**	

Table 4: Music generation performance comparison.

Algorithm	BLEU-4	p-value	MSE	p-value
MLE	0.9210	$< 10^{-6}$	22.38	0.00034
SeqGAN	**0.9406**		**20.62**	

MaskGAN

- GAN to fill in the blank. Encoder - Decoder



https://arxiv.org/pdf/1801.07736.pdf

MaskGAN

- GAN to fill in the blank. Encoder - Decoder

Ground Truth	**Pitch Black was a complete shock to me when I first saw it back in 2000 In the previous years I**
MaskGAN	Pitch Black was a complete shock to me when I first saw it back in 1979 I was really looking forward
	Pitch Black was a complete shock to me when I first saw it back in 1976 The promos were very well
	Pitch Black was a complete shock to me when I first saw it back in the days when I was a
MaskMLE	Black was a complete shock to me when I first saw it back in 1969 I live in New Zealand
	Pitch Black was a complete shock to me when I first saw it back in 1951 It was funny All Interiors
	Pitch Black was a complete shock to me when I first saw it back in the day and I was in

https://arxiv.org/pdf/1801.07736.pdf

Gumbel Softmax

From softmax prob [0.3 0.4 0.1 0.2] -> word index z
Pick randomly based on prob (sampling)

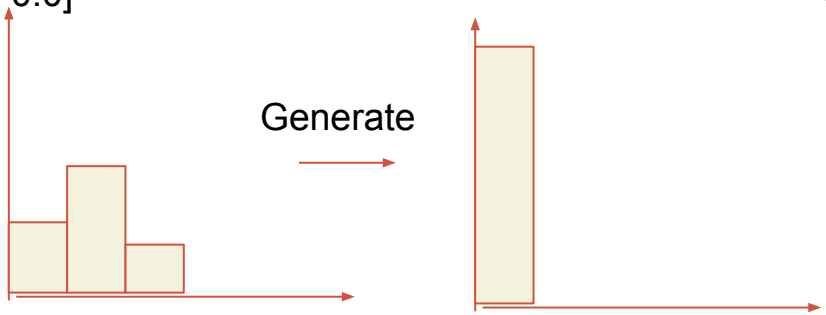
random value generated from Gumbel dist.

$$z = \texttt{one\_hot}\left(\arg\max_i \left[g_i + \log \pi_i\right]\right)$$

prob values from softmax
[1.0, 0.0, 0.0]

index for each word

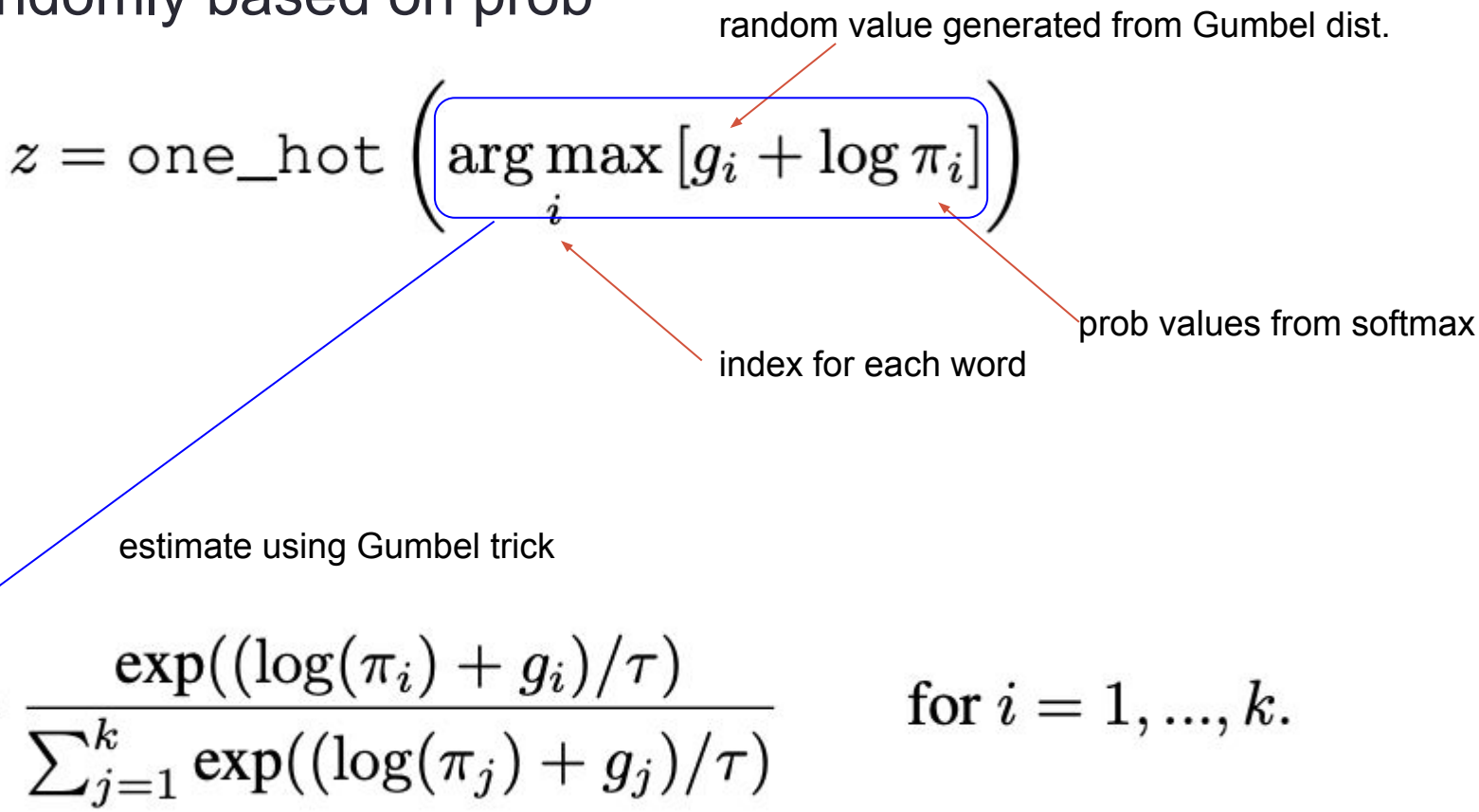
prob values from softmax
[0.3, 0.5, 0.2]



https://arxiv.org/pdf/1611.01144.pdf

Gumbel Softmax

From softmax prob [0.3 0.4 0.1 0.2] -> word index z
Pick randomly based on prob



Gumbel Softmax

From softmax prob [0.3 0.4 0.1 0.2] -> word index z
Pick randomly based on prob

random value generated from Gumbel dist.

$$z = \texttt{one\_hot}\left(\arg\max_i [g_i + \log \pi_i]\right)$$

index for each word

prob values from softmax

Not a one hot

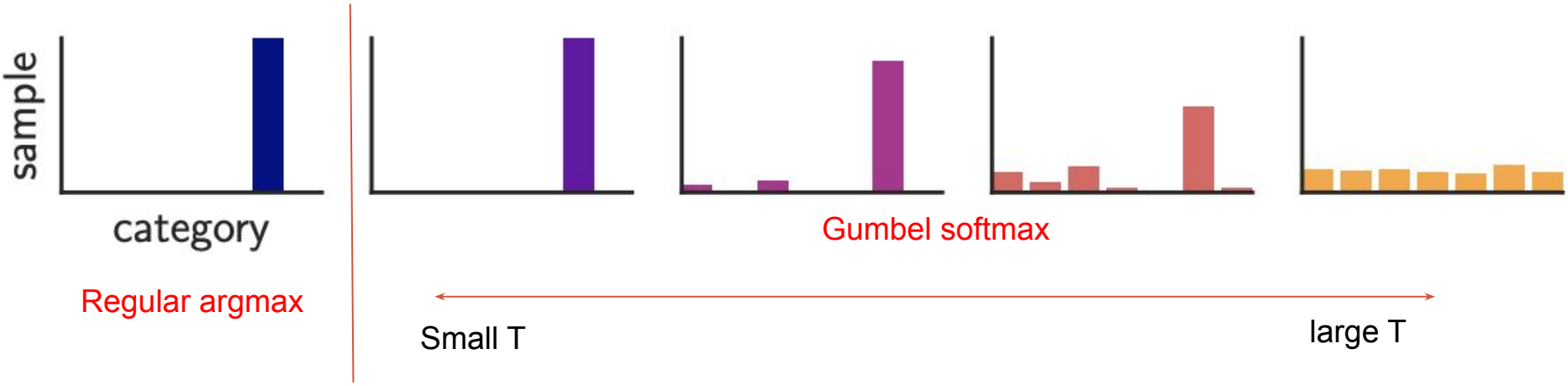
Temperature parameter

$$y_i = \frac{\exp((\log(\pi_i) + g_i)/\tau)}{\sum_{j=1}^{k} \exp((\log(\pi_j) + g_j)/\tau)} \quad \text{for } i = 1, \ldots, k.$$

This rescales the distribution

Gumbel Softmax

Temperature can be scaled
y can be back propagated through



Not a one hot

Temperature parameter

$$y_i = \frac{\exp((\log(\pi_i) + g_i)/\tau)}{\sum_{j=1}^{k} \exp((\log(\pi_j) + g_j)/\tau)} \quad \text{for } i = 1, \ldots, k.$$

This rescales the distribution

GAN readings

- 1) GAN tutorial: https://arxiv.org/pdf/1701.00160.pdf
- 2) WGAN: https://arxiv.org/abs/1701.07875
  - 2.1) Blog explanation: https://www.alexirpan.com/2017/02/22/wasserstein-gan.html Read 2) and 2.1) together section by section.
- 3) WGAN-GP: https://arxiv.org/abs/1704.00028
- 4) On how GANs are hard to train stil: https://arxiv.org/abs/1711.10337

GAN notes

Unlike computer vision, GAN is rarely used in NLP

However, it is common for adversarial loss to be used jointly with typical loss in NLP

For most text generation, CE is usually enough (require large data)

Overview

Techniques
- Transformer
- Beam search
- GAN for text

Applications
- Grammatical Error Correction

Next time
- More transformers