

Natural Language Processing II

Jiří Materna





@mlcollegecom



@mlcollegecom



About me

- Ph.D. in Natural Language Processing and Artificial Intelligence at Masaryk University
- 10 years at seznam.cz (last 8 years as Head Of Research)
- Founder and co-organizer of ML Prague
- Mentor at StartupYard
- ML Freelancer and consultant

www.mlguru.com

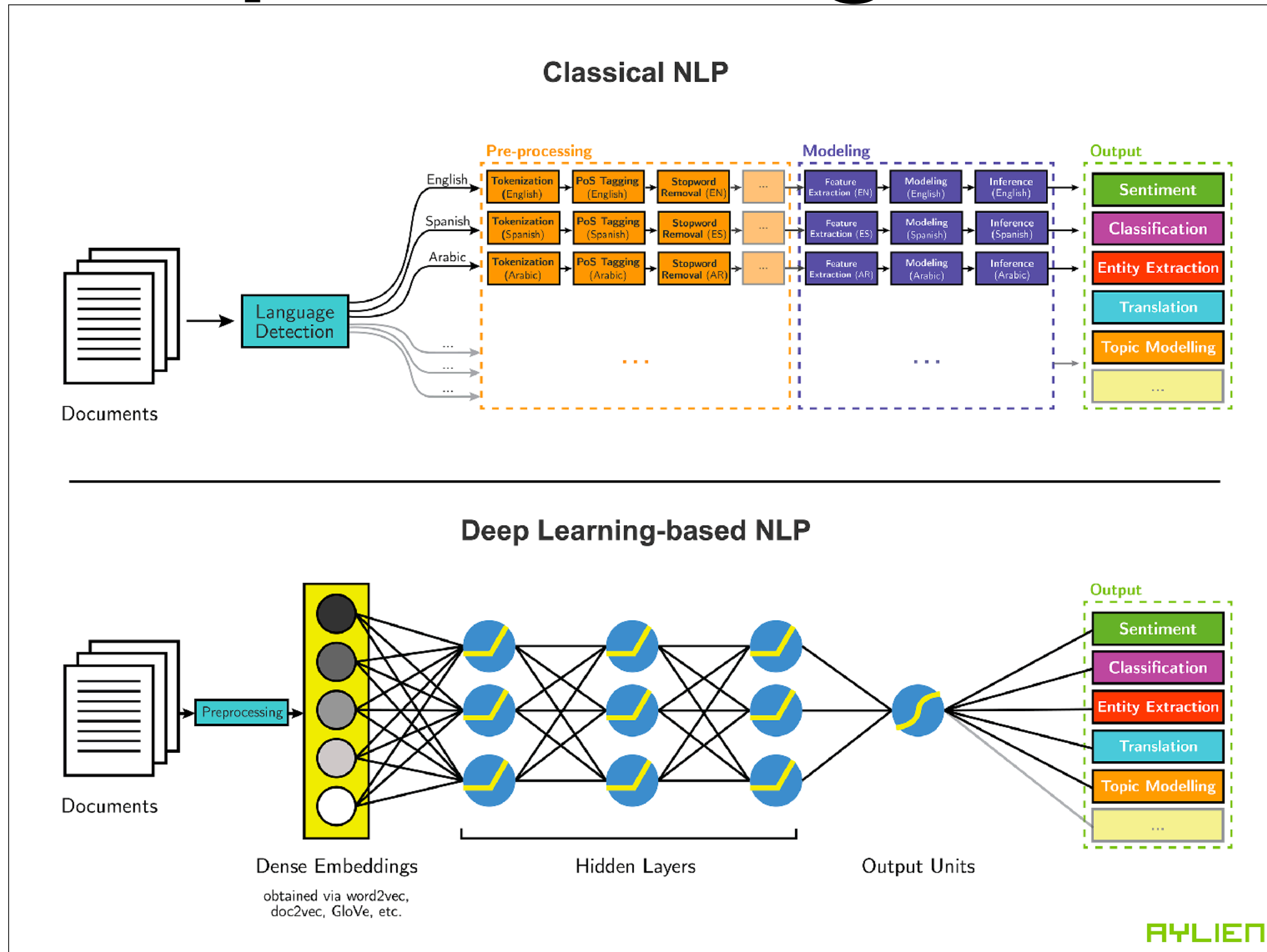
www.mlprague.com

www.mlcollege.com

Outline

- Preprocessing for deep learning in NLP
- Recurrent neural networks
- Word embeddings and word2vec
- The Skip-gram model
- Text classification with word embeddings
- Subword tokenization
- LSTM and GRU
- Attention is all you need
- Transformers (GPT3, BERT, XLNET)
- Practical task on classification using BERT

Deep Learning in NLP



Encoding and Unicode

ASCII

H e l l o

48 65 6c 6c 6f

Unicode

H e l l o ☺

00000048 00000065 0000006c 0000006c 0000006f 0000263a

Encoding and Unicode

UTF-8

H e l l o ☺

48 65 6c 6c 6f e298ba

UTF-16

H e l l o ☺

0048 0065 006c 006c 006f 263a

Unicode normalization

NFD (Normalization Form Canonical Decomposition)

NFC (Normalization Form Canonical Composition)

NFKD (Normalization Form Compatibility Decomposition)

NFKC (Normalization Form Compatibility Composition)

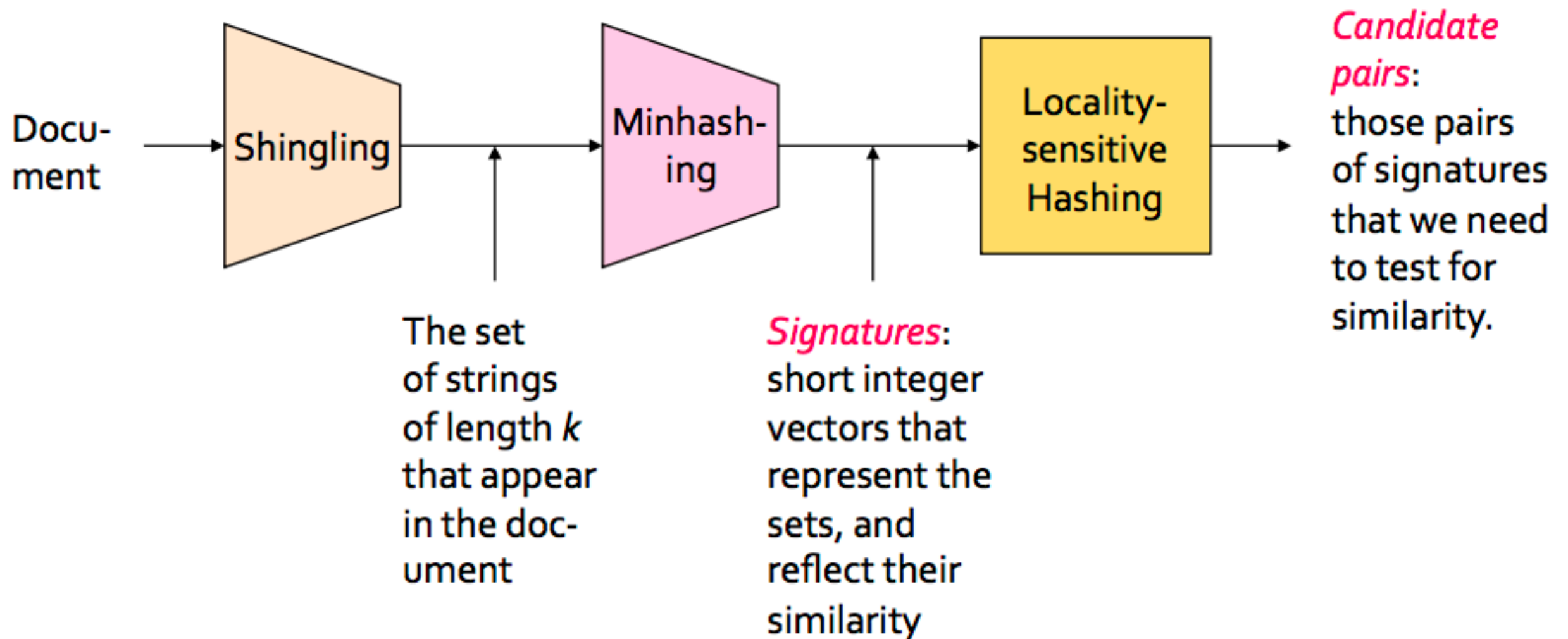
Source		NFD		NFC		NFKD		NFKC
fi FB01	:	fi FB01		fi FB01		f i 0066 0069		f i 0066 0069
2 ⁵ 0032 2075	:	2 5 0032 2075		2 5 0032 2075		2 5 0032 0035		2 5 0032 0035
ḟ 1E9B 0323	:	f ͡ ͡ 017F 0323 0307		ḟ ͡ 1E9B 0323		s ͡ ͡ 0073 0323 0307		ṡ 1E69

Unicode normalization in Python 3

```
>>> aa = b'\xc4\x81'.decode('utf8')
>>> bb = b'a\xcc\x84'.decode('utf8')
>>> aa
'ā'
>>> bb
'ā'
>>> aa == bb
False
>>> import unicodedata as ud
>>> aa == ud.normalize('NFC',bb)
True
```

Near deduplication

Locality-sensitive hashing

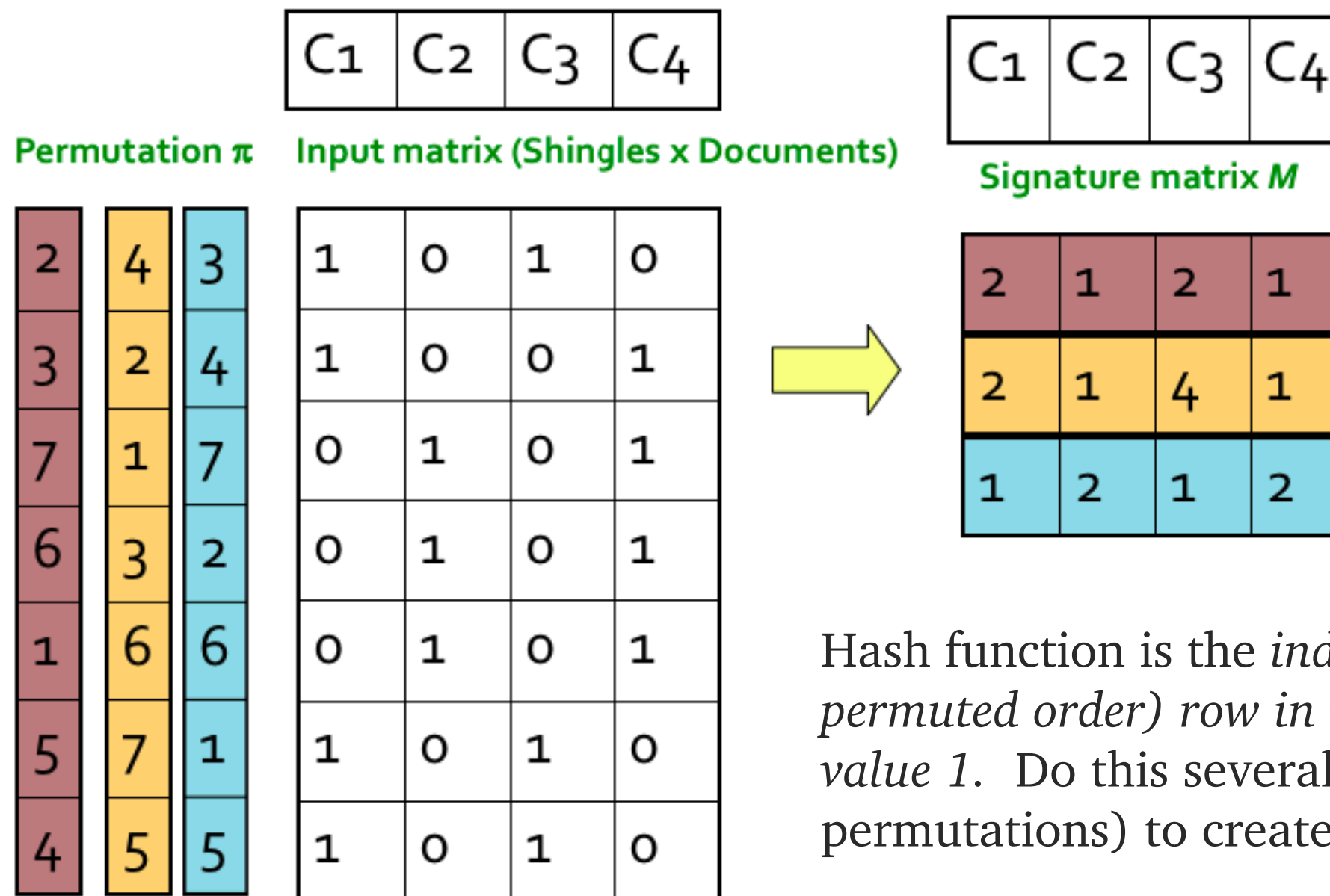


Set of shingles (n-grams) as document representation

		Documents			
Shingles	1	1	1	1	0
	2	1	1	0	1
	3	0	1	0	1
	4	0	0	0	1
	5	1	0	0	1
	6	1	1	1	0
	7	1	0	1	0

Near deduplication

MinHashing signatures



Jaccard similarity and MinHashing signatures

C ₁	C ₂	C ₃	C ₄

Signature matrix *M*

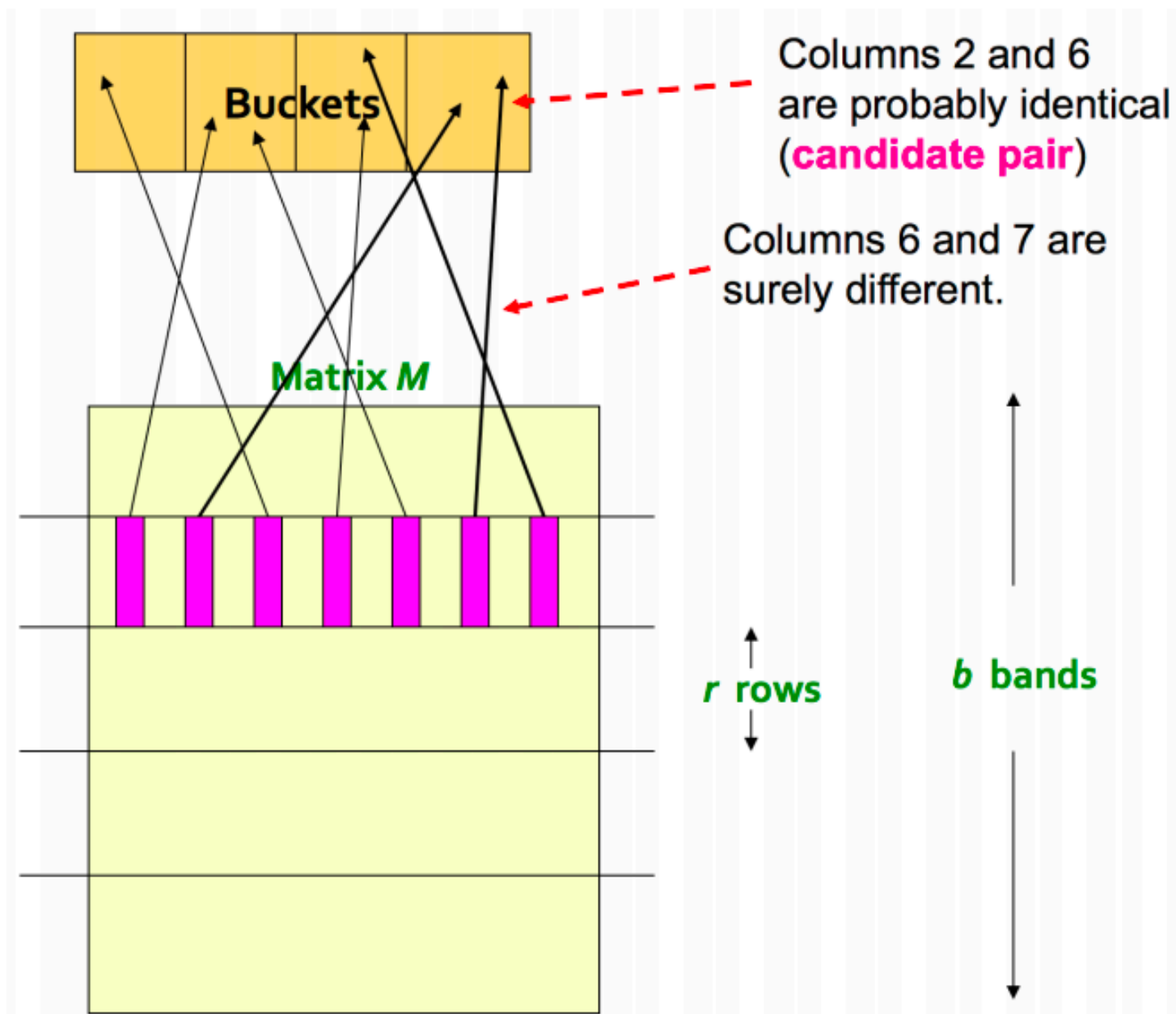
2	1	2	1
2	1	4	1
1	2	1	2

The similarity of the signatures is the fraction of the min-hash functions (rows) in which they agree. So the similarity of signature for C₁ and C₃ is 2/3 as 1st and 3rd row are same.

Claim: $P[h_{\pi}(C_1) = h_{\pi}(C_2)] = \text{sim}(C_1, C_2)$

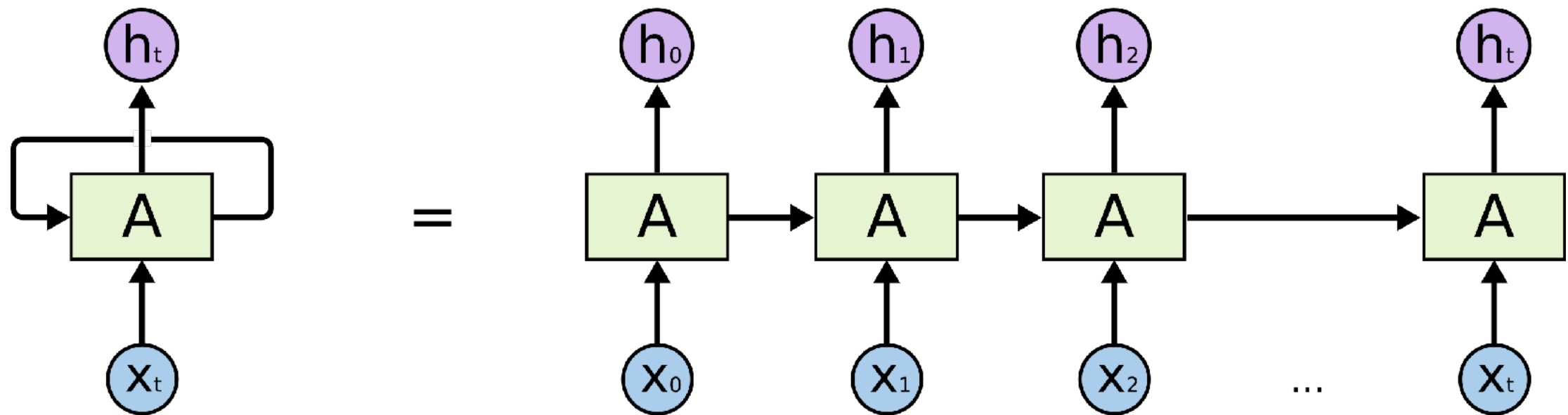
Near deduplication

Locality-sensitive hashing



Recurrent Neural networks

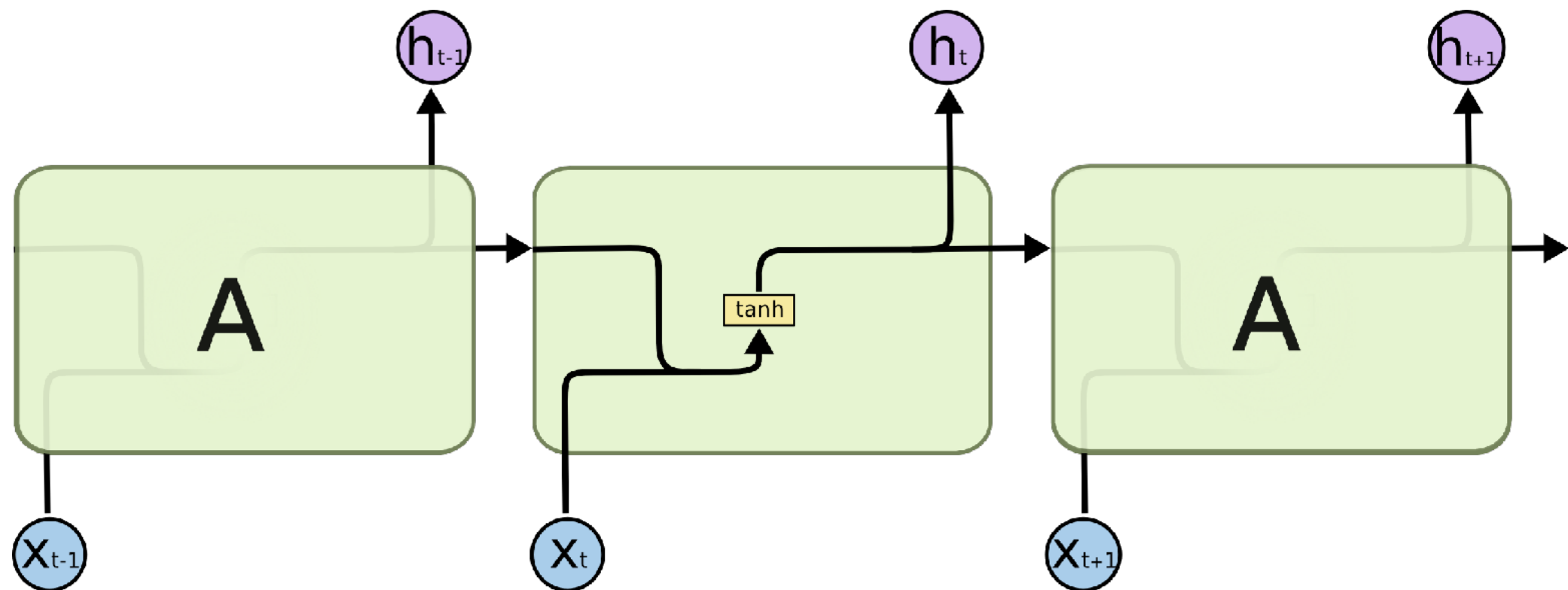
1/2



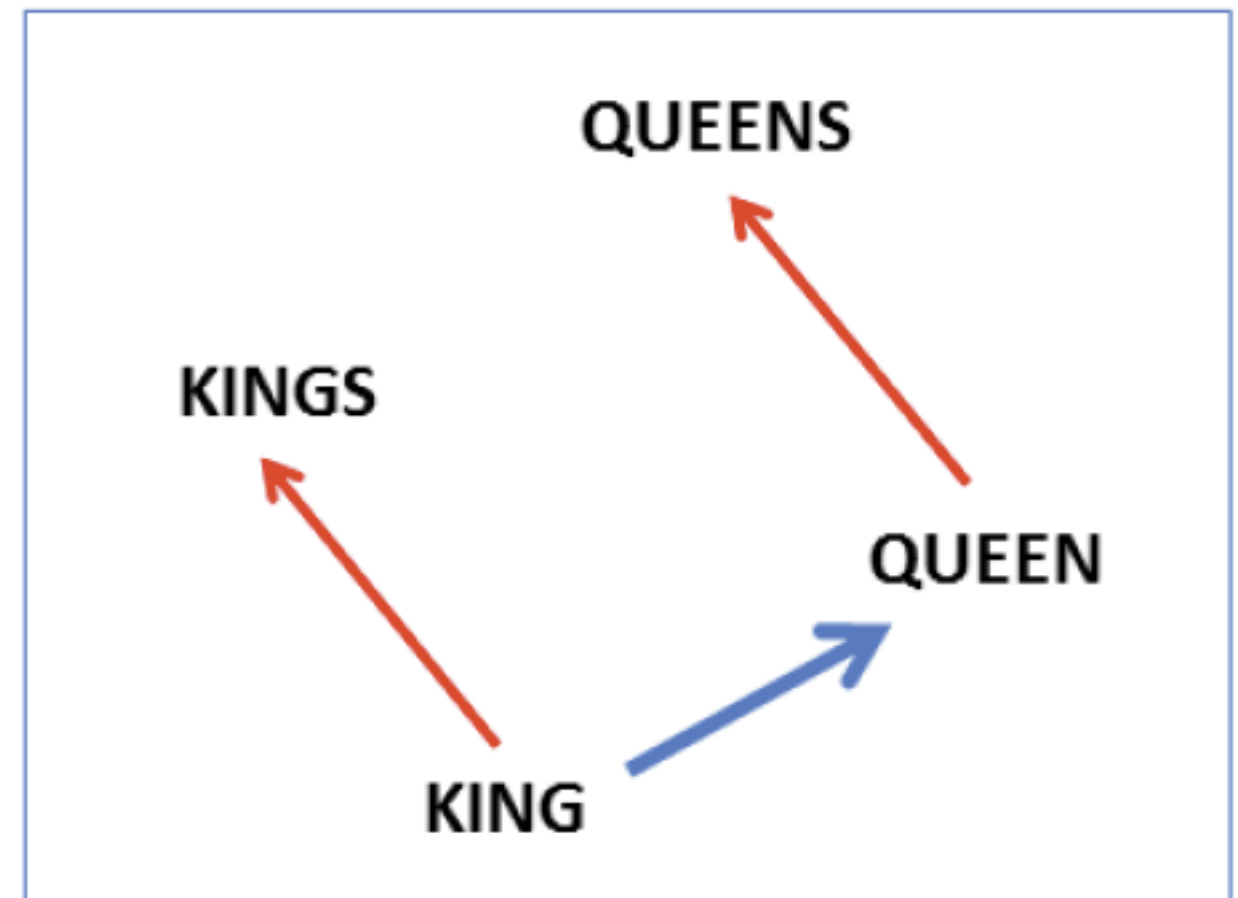
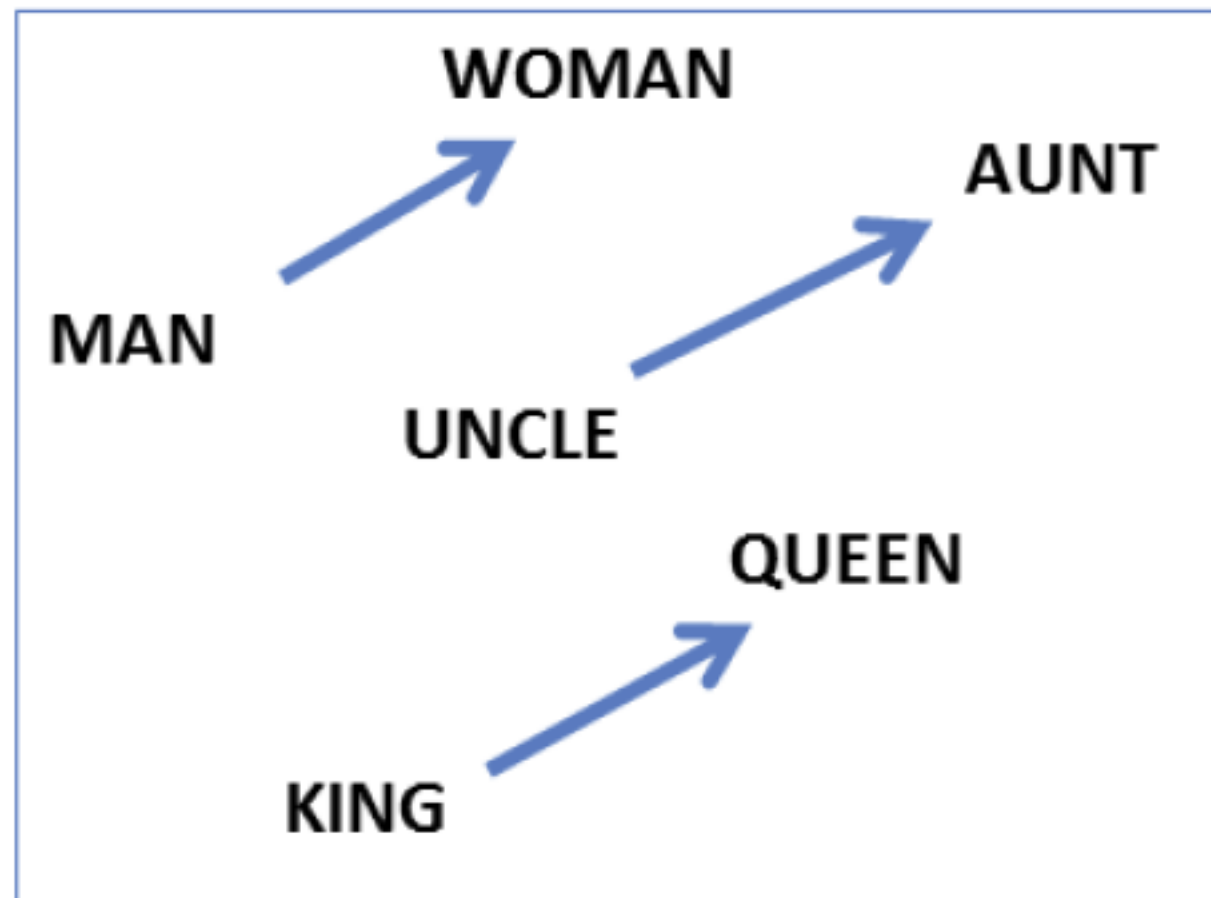
source: <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

Recurrent Neural Networks

2/2



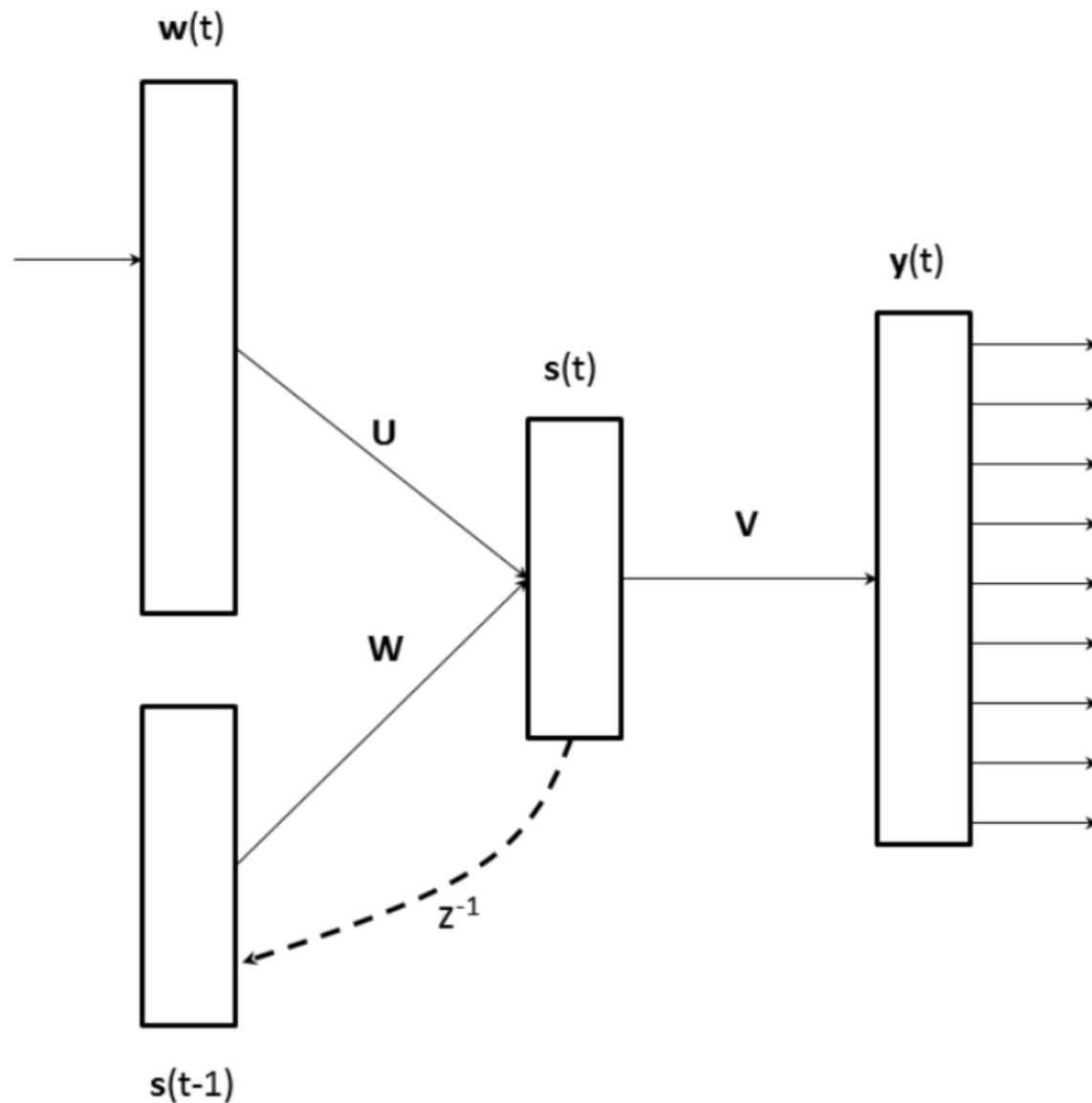
word2vec



king is to **kings** as **queen** to ?.

$$v(\mathbf{kings}) - v(\mathbf{king}) = v(\mathbf{queens}) - v(\mathbf{queen})$$

Recurrent Neural Network Language Modeling Toolkit

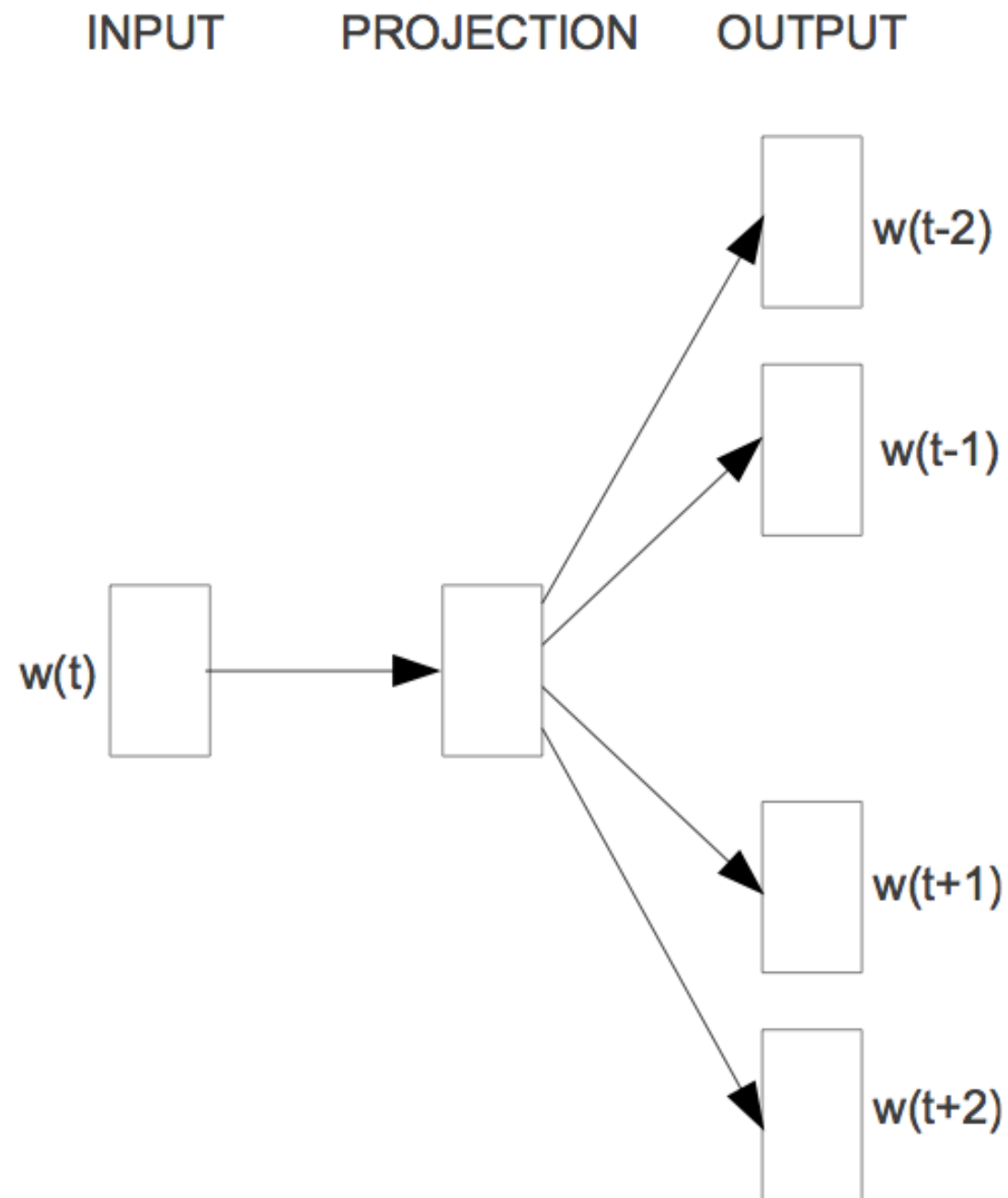


$$\mathbf{s}(t) = f(\mathbf{U}\mathbf{w}(t) + \mathbf{W}\mathbf{s}(t-1))$$

$$\mathbf{y}(t) = g(\mathbf{V}\mathbf{s}(t)),$$

$$f(z) = \frac{1}{1 + e^{-z}}, \quad g(z_m) = \frac{e^{z_m}}{\sum_k e^{z_k}}.$$

The Skip-gram model



Skip-gram improvements

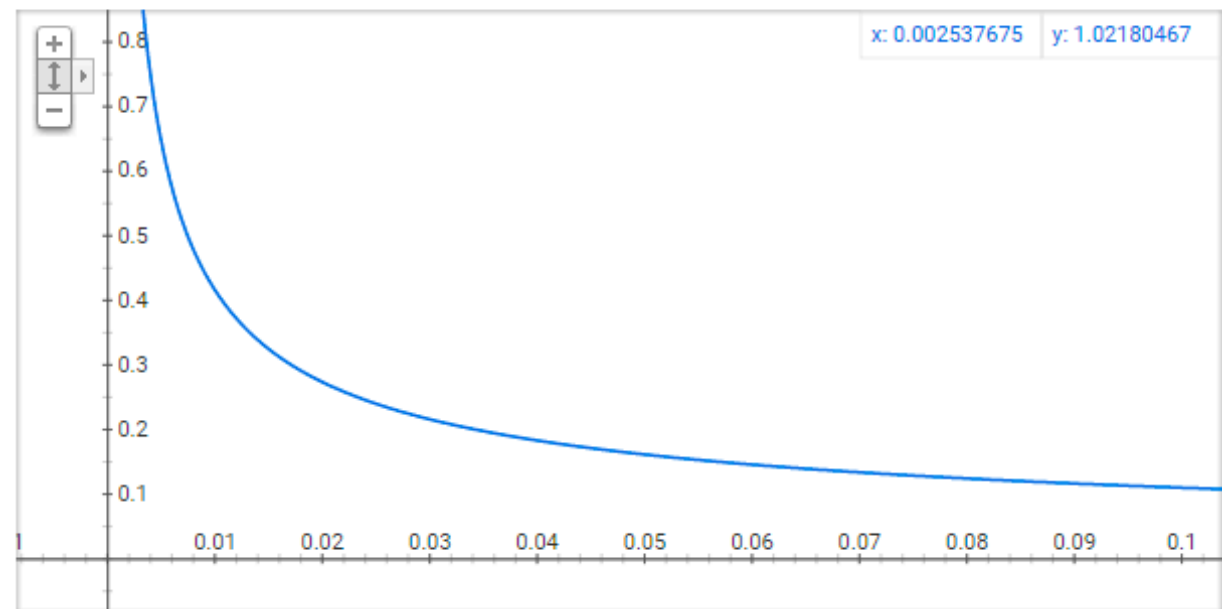
Subsampling frequent inputs

$$P(w_i) = \left(\sqrt{\frac{z(w_i)}{0.001}} + 1 \right) \cdot \frac{0.001}{z(w_i)}$$

$z(w)$ Relative frequency of word w

$P(w)$ Probability of keeping word w

Graph for $(\sqrt{x/0.001}+1)*0.001/x$



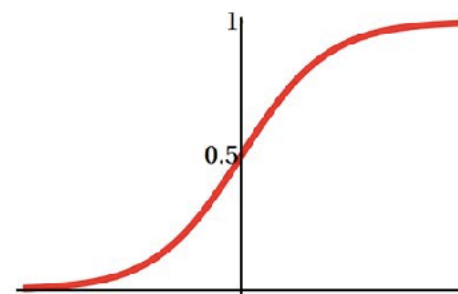
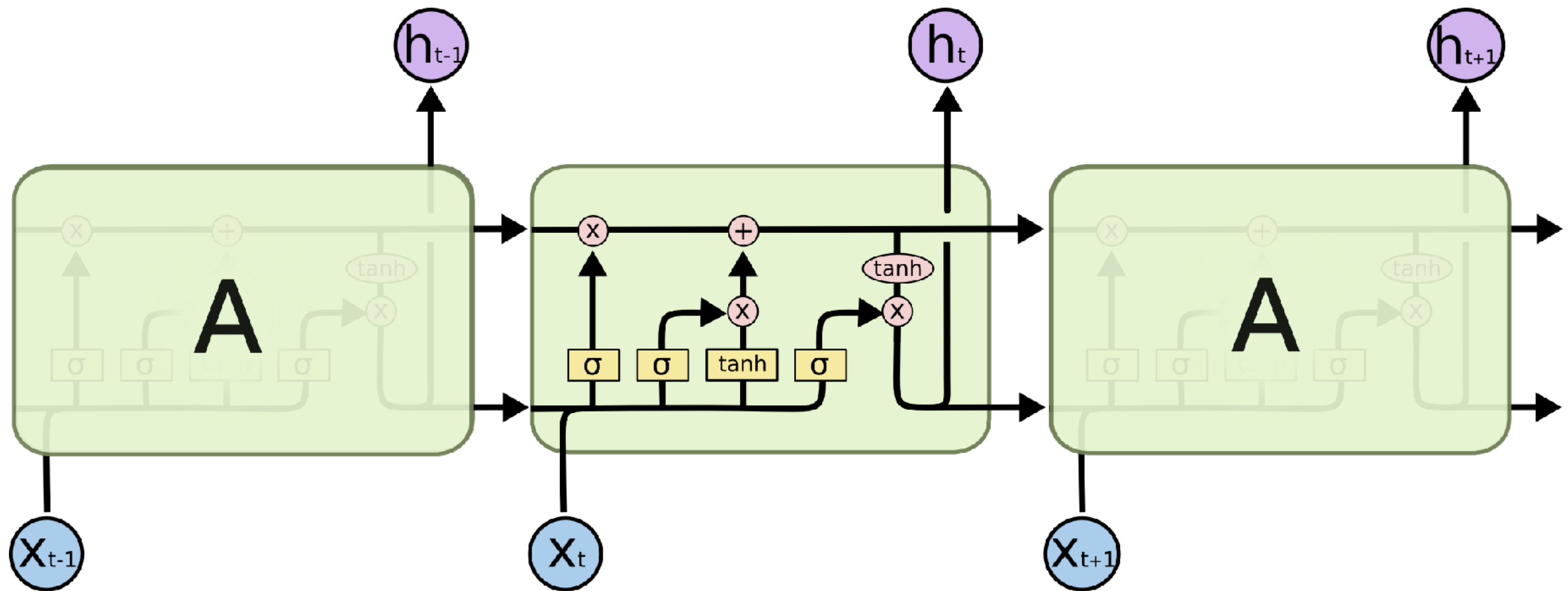
Negative sampling

We select only 5-20 negative samples in the loss function.
The probability of picking a word w is given by $z(w)$.

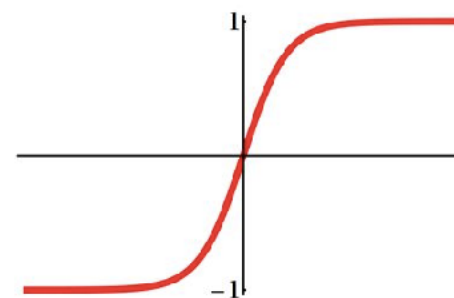
Experiments with word2vec

01-Review-classification-w2v-assignment.ipynb

Long Short-Term Memory



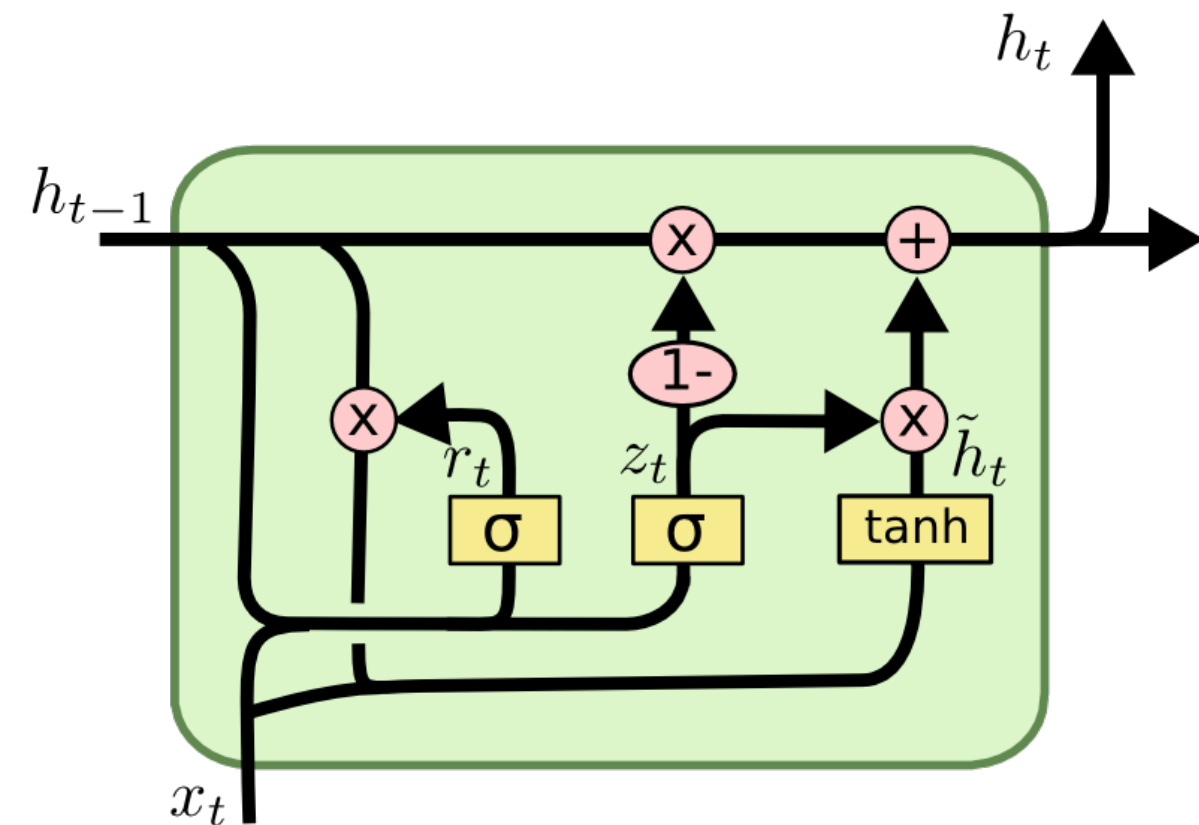
sigmoid



tanh

Zdroj: <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

Gated Recurrent Unit



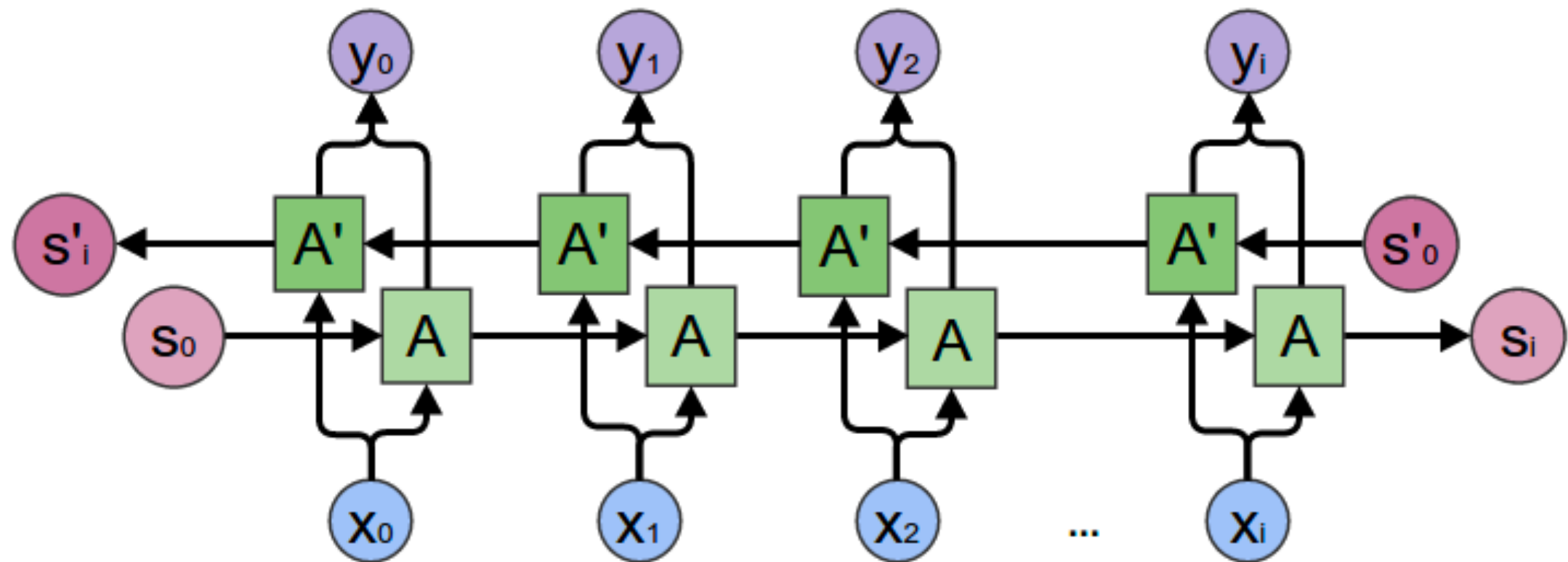
$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$$

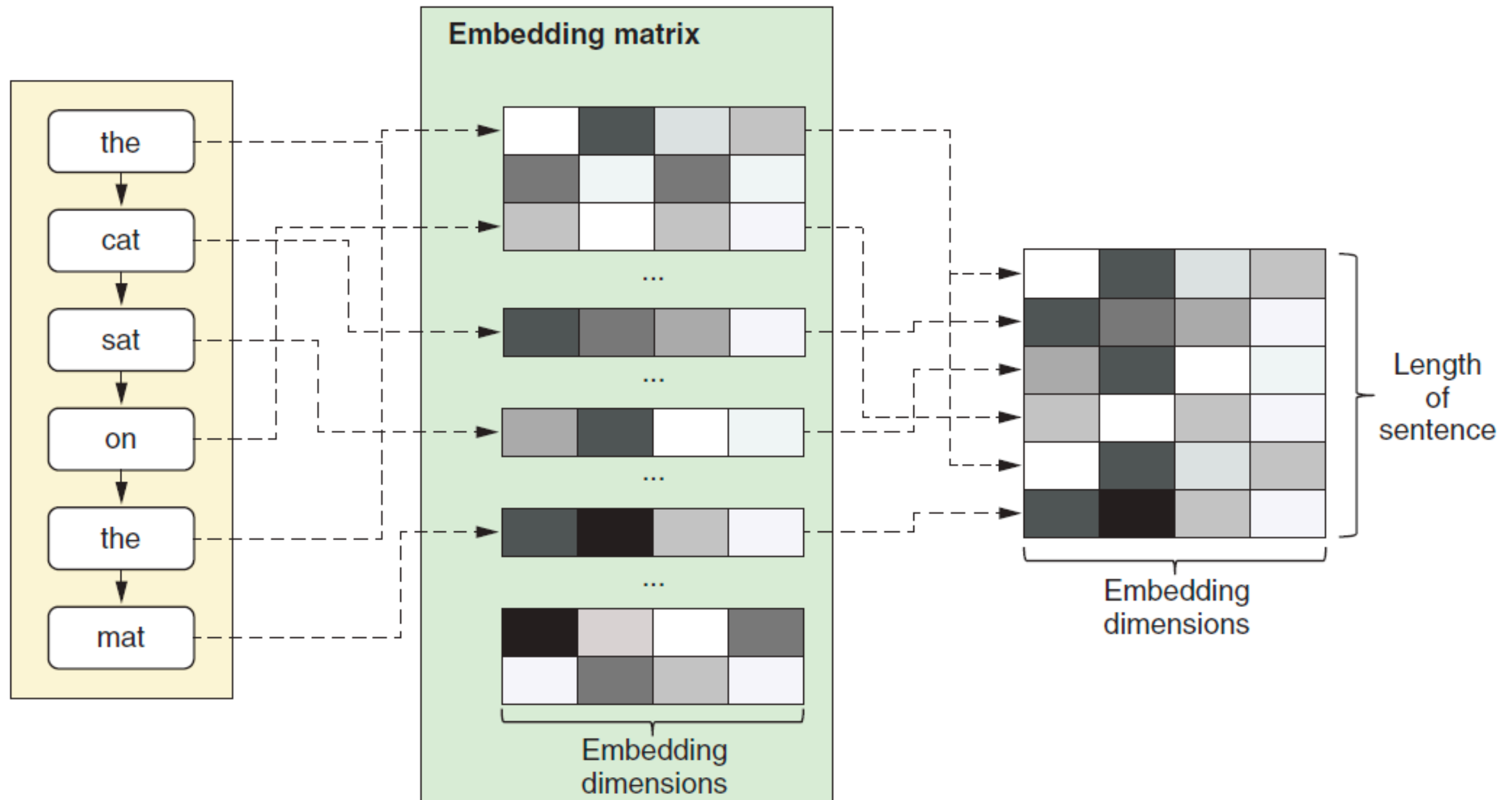
$$\tilde{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t])$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Bidirectional recursive layer in Keras



Embedding layer in Keras



Text classification with bidirectional LSTM

02-Review-classification-LSTM.ipynb

Traditional tokenization

NLTK tokenizers

```
>>> from nltk.tokenize import word_tokenize #simple
>>> from nltk.tokenize.moses import MosesTokenizer #enables detokenization
>>> from nltk.tokenize import ToktokTokenizer #fast
>>>
>>> moses = MosesTokenizer()
>>> toktok = ToktokTokenizer()
>>>
>>> text = "Welcome to Machine Learning College."
>>> print(word_tokenize(text))
>>> print (moses.tokenize(text))
>>> print (toktok.tokenize(text))
['Welcome', 'to', 'Machine', 'Learning', 'College', '.']
['Welcome', 'to', 'Machine', 'Learning', 'College', '.']
['Welcome', 'to', 'Machine', 'Learning', 'College', '.']
```

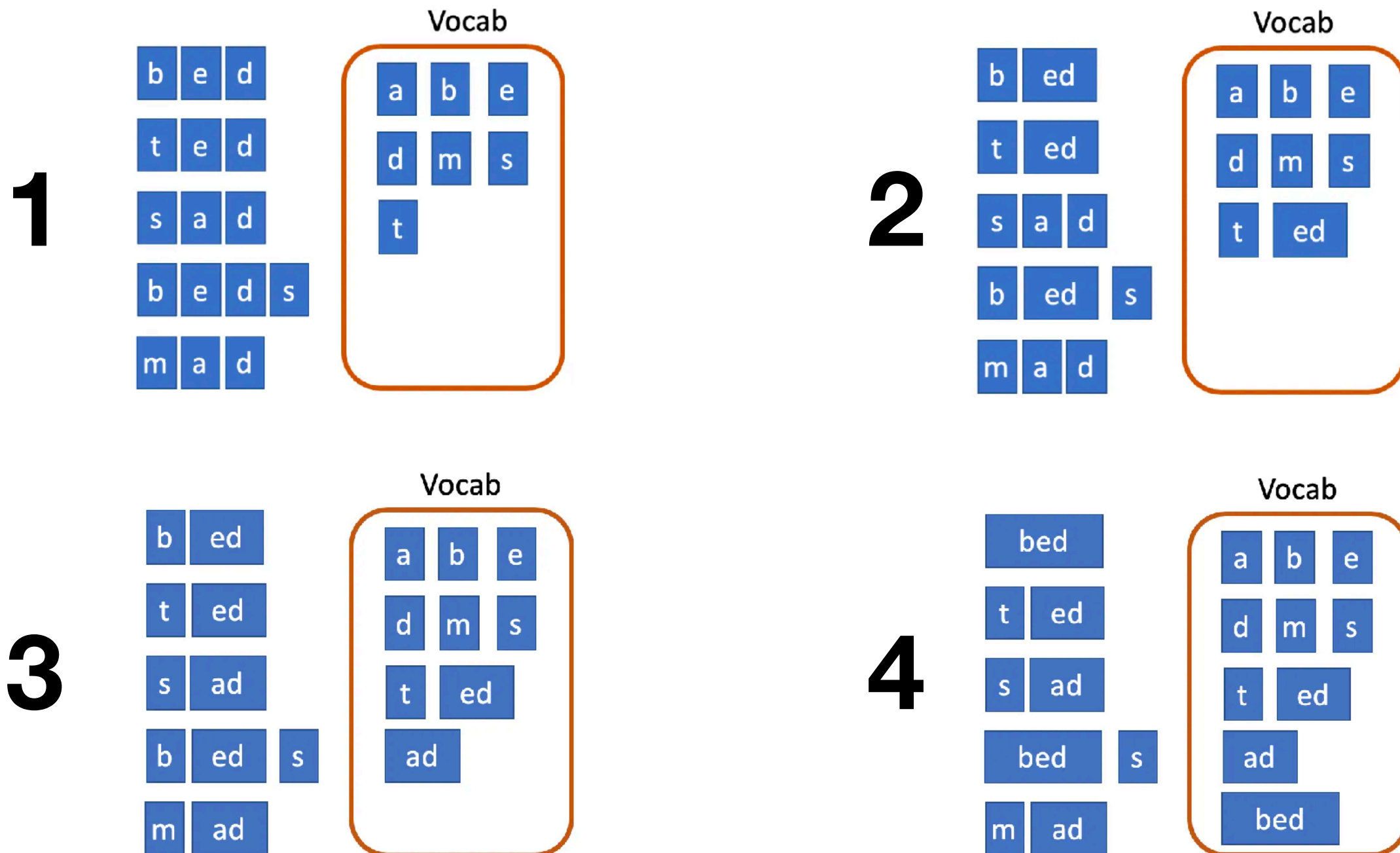
Traditional tokenization

SpaCy tokenizer

```
>>> import spacy
>>> sp = spacy.load('en_core_web_sm')
>>> tokens = sp("Welcome to Machine Learning College.")
>>>
>>> [word.text for word in tokens]
['Welcome', 'to', 'Machine', 'Learning', 'College', '.']
```

Subword tokenization

Byte-pair encoding



Subword tokenization

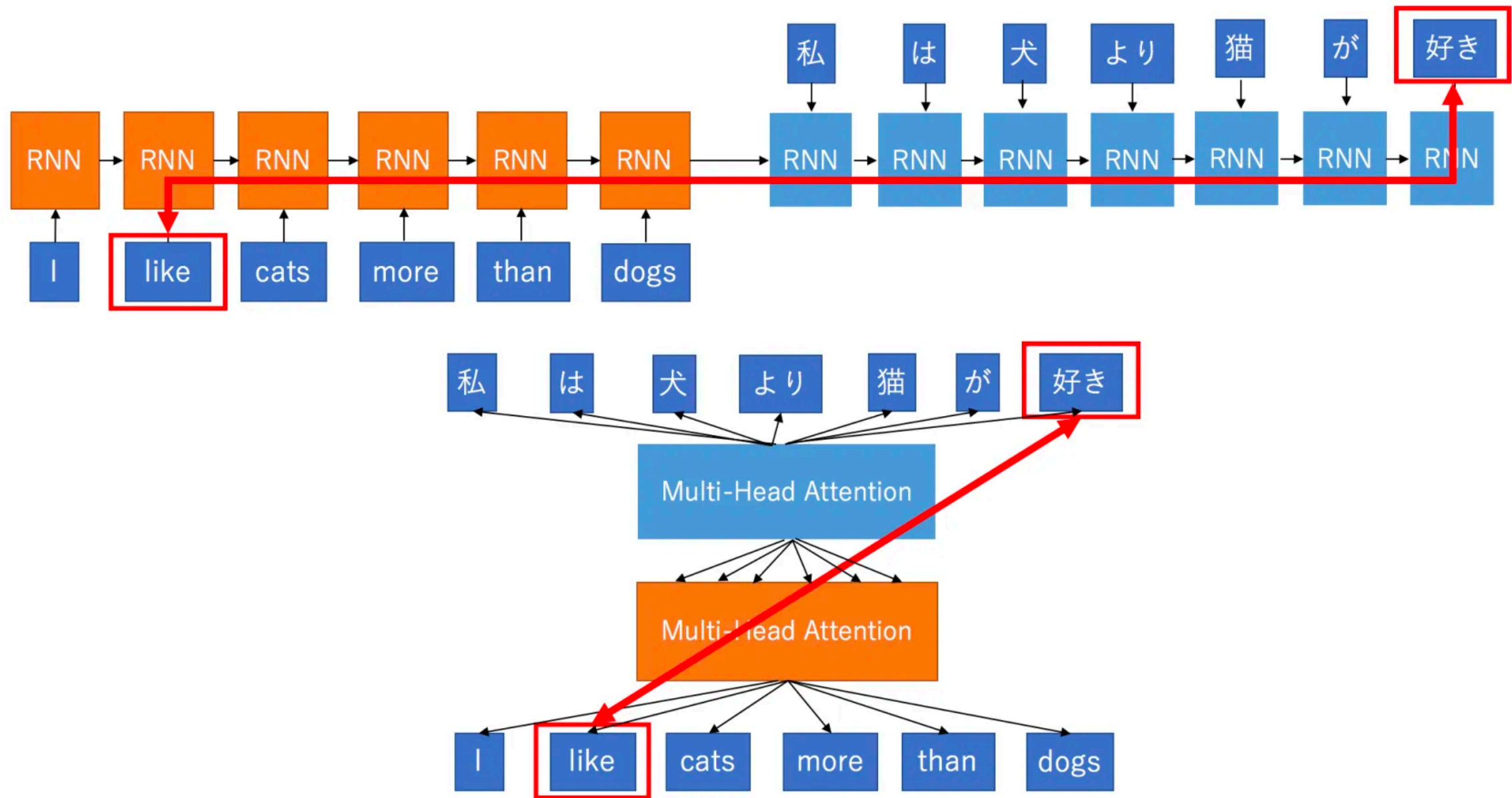
Wordpiece and sentencepiece tokenization

Merges bigrams with maximum mutual information instead of maximum frequency.

$$I(x, y) = \log \left(\frac{p(x, y)}{p(x) p(y)} \right)$$

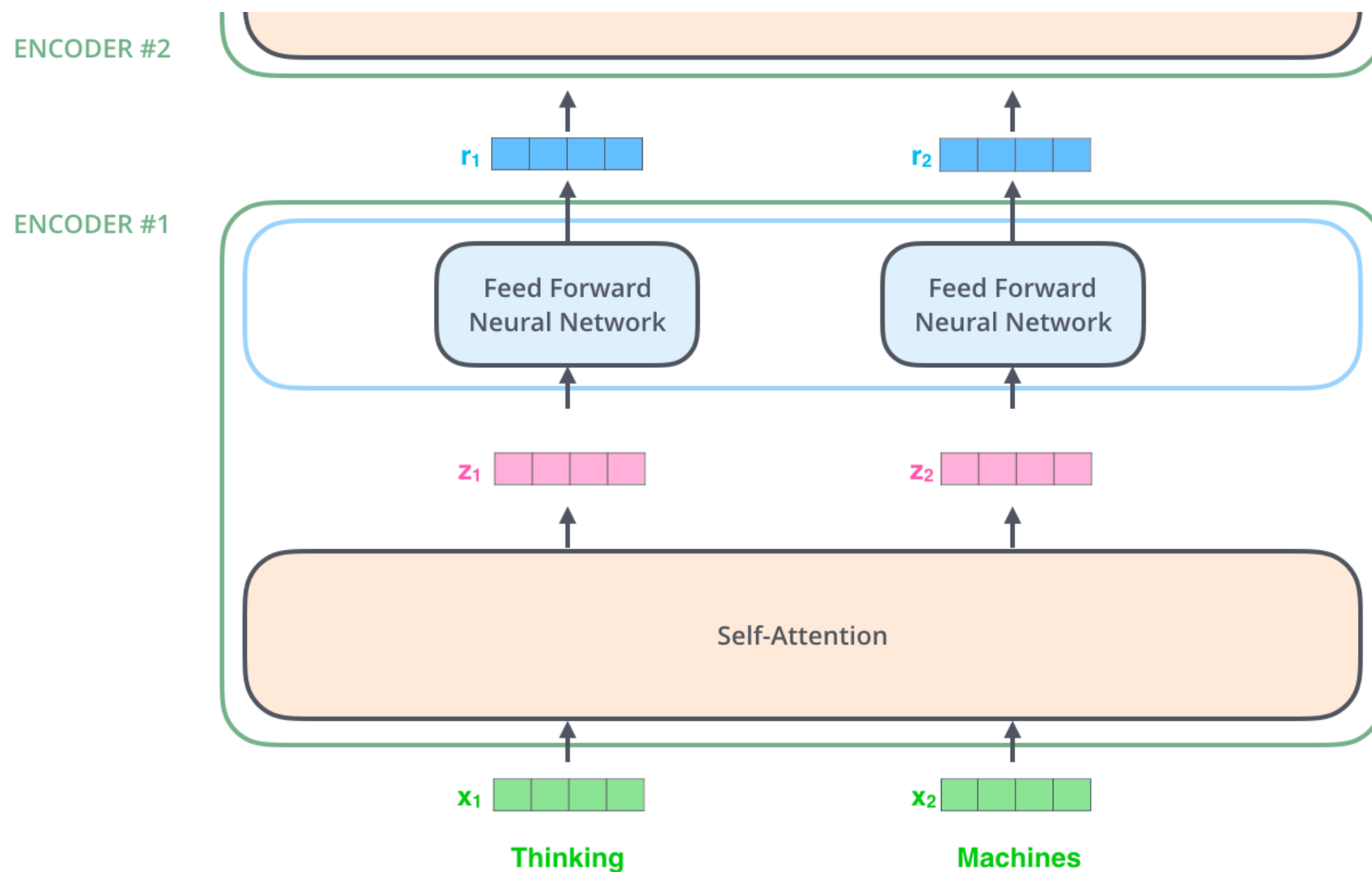
playing -> play, ##ing

RNN vs. Transformer

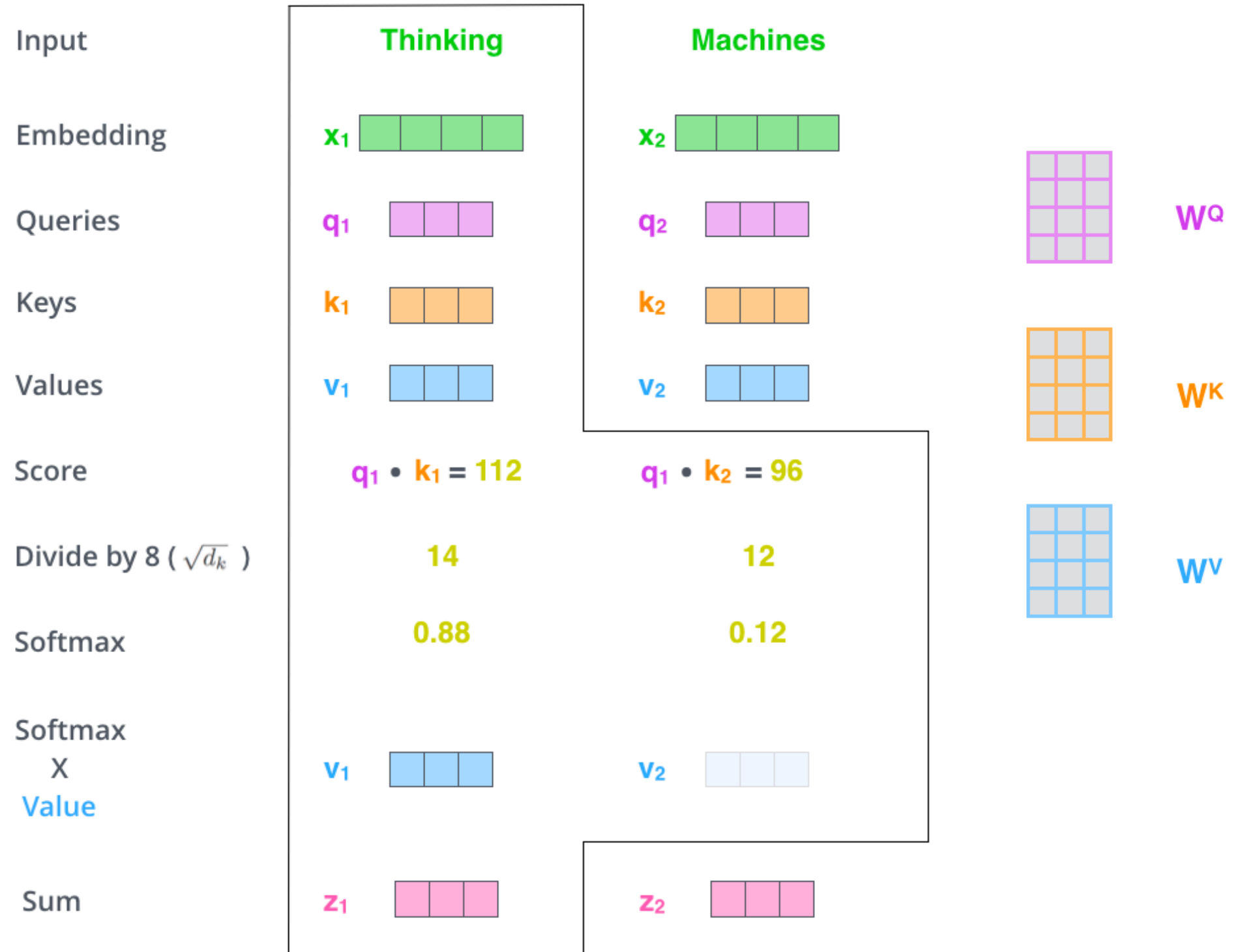


source: www.mlexplained.com

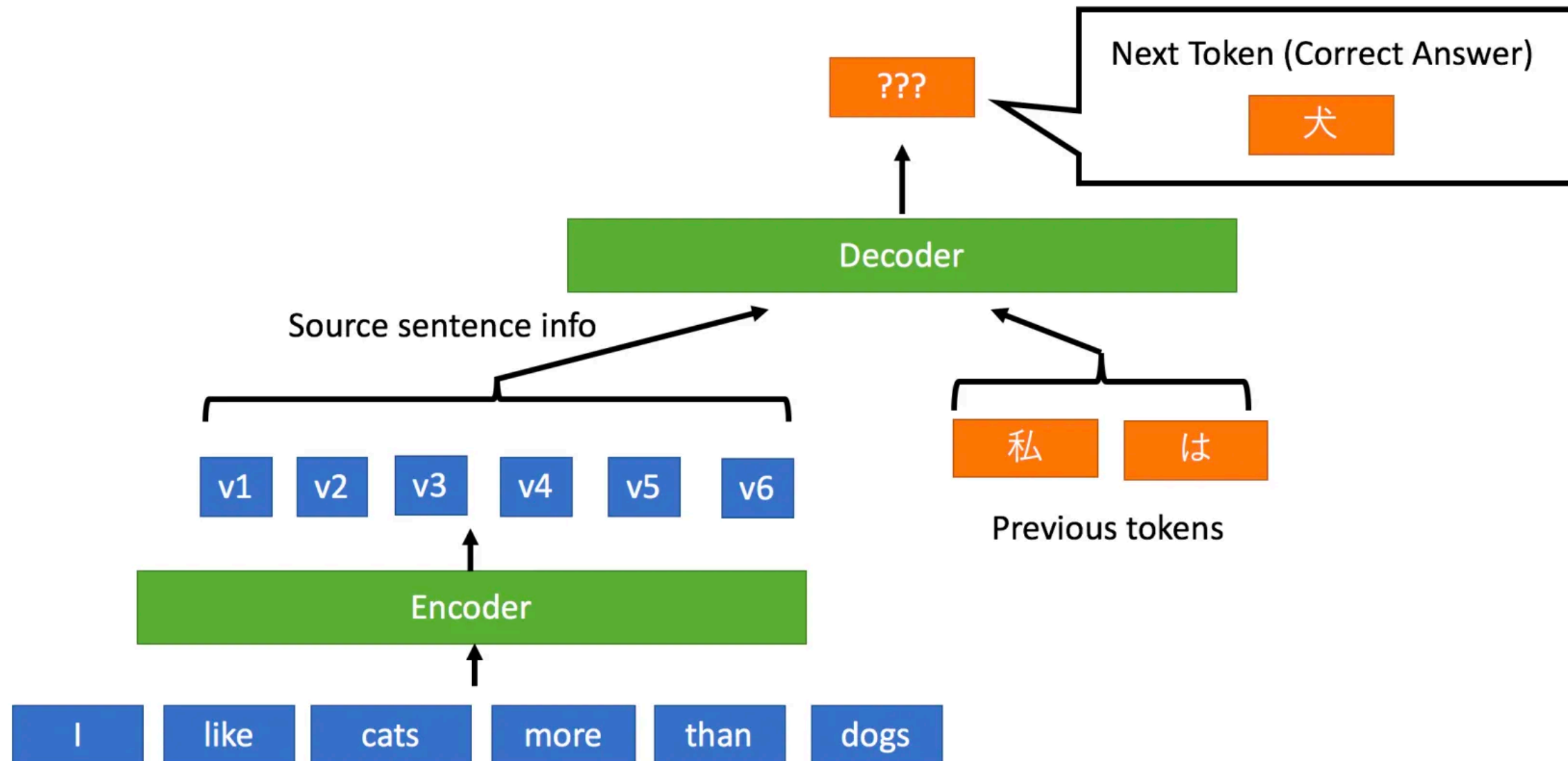
Attention is all you need



Self-attention



Translation with Transformers



GPT-2 Language model

Donald Trump told...

GPT-2 Language model

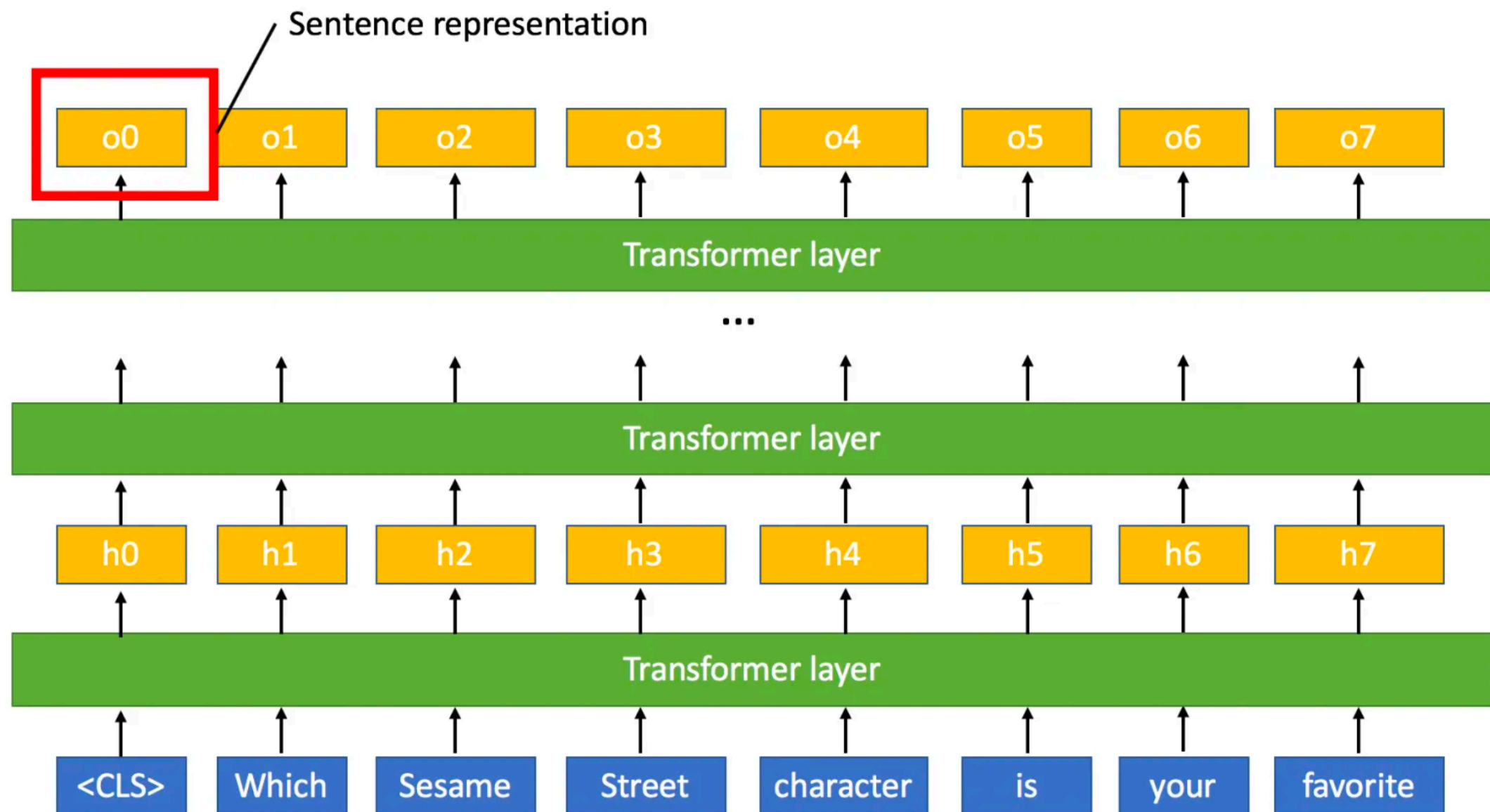
Donald Trump told the Times he is preparing a "major speech" on his economic plans, but did not provide details on what it will entail.

“I’m getting ready for the speech. And I will have a major speech on Tuesday.” Trump said during an interview in the White House residence.

GPT-3: <https://openai.com/blog/openai-api/>

BERT

(classification)



Text classification using BERT

03-Review-classification-BERT.ipynb

Thank you for your attention

e-mail: jiri@mlcollege.com

Web: www.mlcollege.com

Twitter: @JiriMaterna

Facebook: <https://www.facebook.com/maternajiri>

LinkedIn: <https://www.linkedin.com/in/jirimaterna/>