

Глубинное обучение

Лекция 7: Нейросети в задачах обработки текстов

Лектор: Максим Рябинин
Исследователь, Yandex Research

Программа ML Residency: yandex.ru/yaintern/research_ml_residency

ФКН ВШЭ, 2022



Виды задач для нейросетей

- Очень сильно упрощаем: текст – последовательность токенов
- Классификация последовательностей
 - Примеры: определение темы, sentiment analysis
 - Учитывать последовательность или нет?
 - Bag-of-words, RNN, transformer
- Разметка последовательности
 - Примеры: определение частей речи, chunking
 - Локальный классификатор, RNN, CRF, RNN-CRF (BiLSTM-CRF), transformer
- Последовательности в последовательность (разной длины!)
 - Примеры: машинный перевод, аннотация (summarization), диалоги
 - Авторегрессионные модели: seq2seq (+attention), ByteNet, transformer и т.д.
 - Неавторегрессионные модели
- Генерация последовательностей
 - Примеры: описание изображения (captioning)
 - Авторегрессивные и неавторегрессивные модели

План лекции

- Непрерывные представления слов (embedding)
 - word2vec, FastText
- Обработка последовательностей
 - Seq2seq
 - Seq2seq + attention
 - Transformer
- Контекстно-зависимые представления (предобучение)
 - ELMo, BERT и его друзья

Как вставить текст в нейросеть?

Непрерывные представления слов (word embeddings)

- Позволяют строить непрерывные представления дискретных объектов
- Непрерывные представления – это способ поместить текст в нейросеть
- Представление – вектор по индексу (токена: слова, символы, n-граммы)
- Представления могут обучаться совместно с моделью
- Предобученные представления
 - Обучены на больших корпусах текстов
 - Обучены без ручной разметки (self-supervision)
 - Freeze, fine-tune, train from scratch?

Представления word2vec (skipgram)

- Обучение на предсказании контекста по слову
 - Обучение на корпусе текстов без разметки (self supervision)
- Вспомогательная задача:
 - Предсказываем каждое слово из контекста отдельно

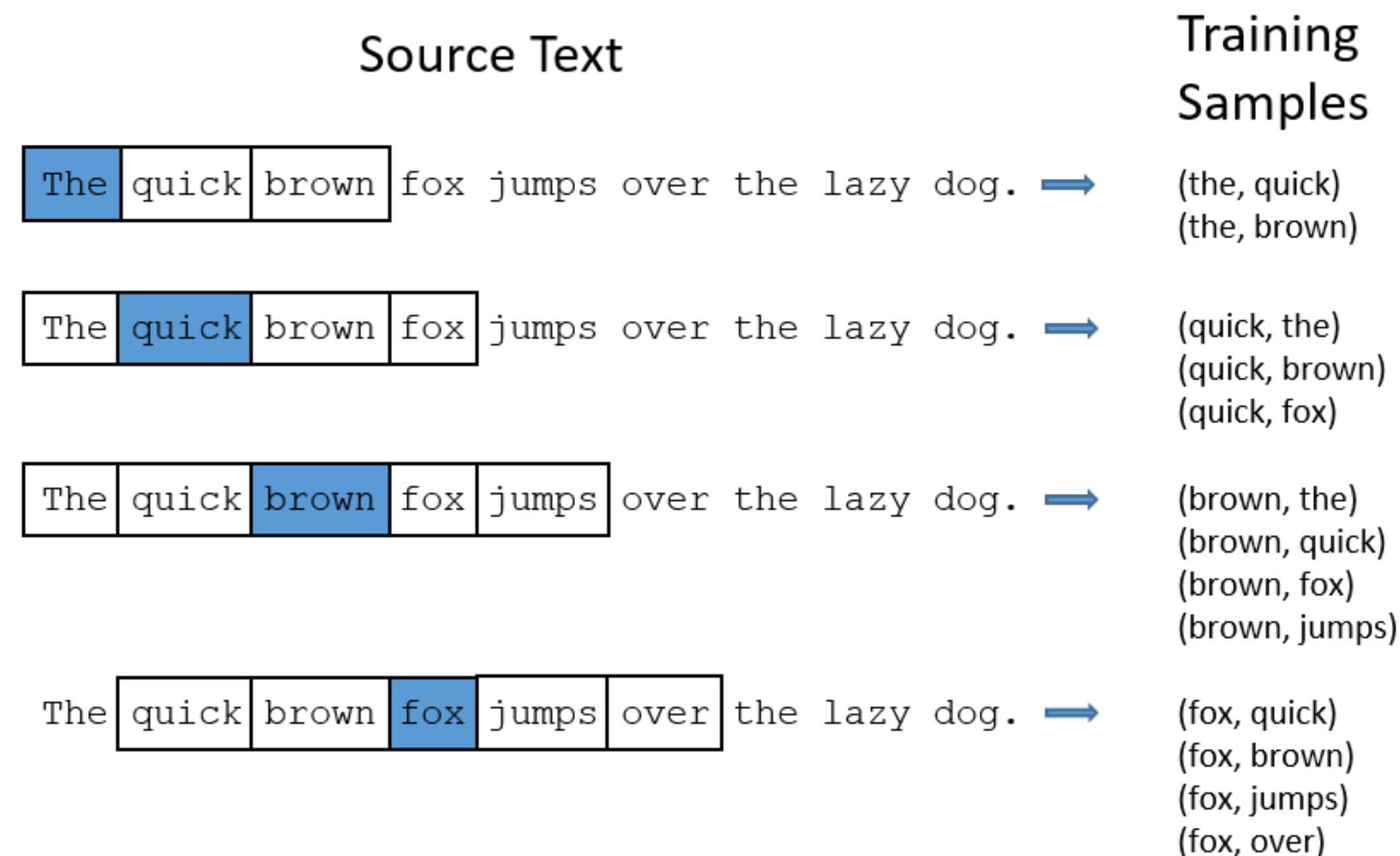


image credit:
**Chris
McCormick**

[Mikolov et al., 2013]

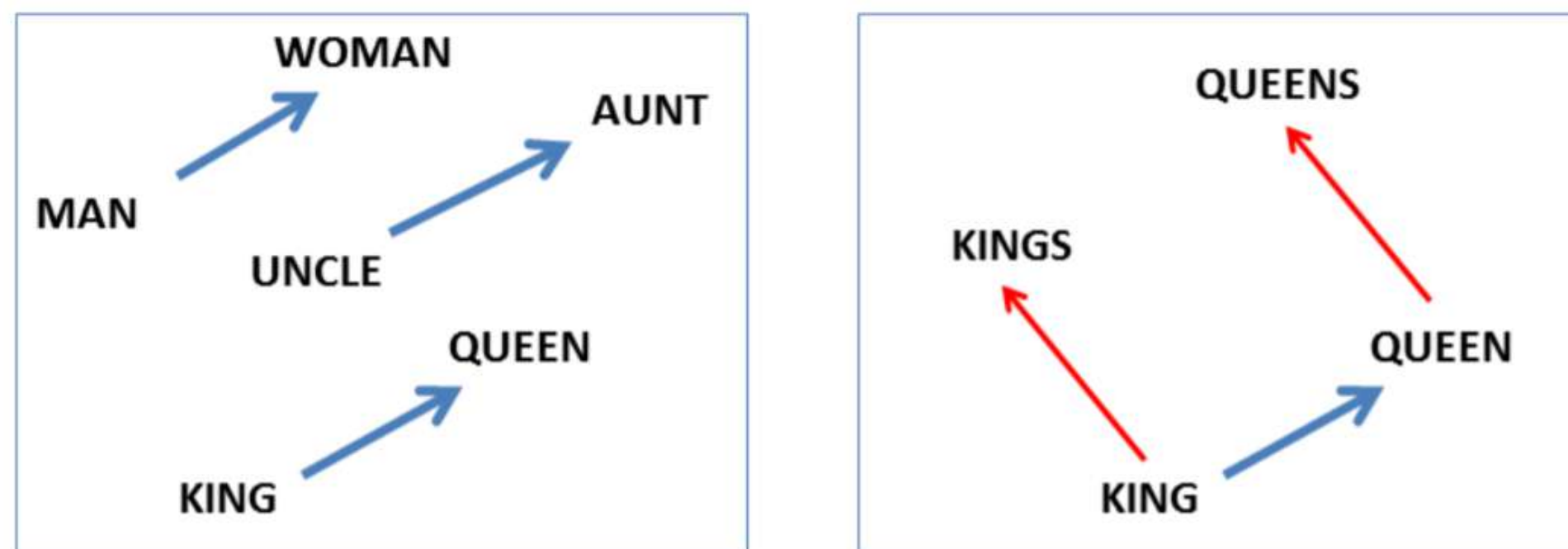
Представления word2vec (skipgram)

- Обучение на предсказании контекста по слову
 - Обучение на корпусе текстов без разметки
- Предсказываем каждое слово из контекста отдельно
 - Текущее слово w ; слово из контекста v
 - Для каждого слова – 2 представления (in, out) $in_w^T out_v$
 - Совместимость – скалярное произведение
 - Полезные – представления in
 - Модель с softmax $P(v | w, \theta) = \frac{\exp(in_w^T out_v)}{\sum_{v'} \exp(in_w^T out_{v'})}$
 - Медленная нормировка
 - Обычное решение – Noise Contrastive Estimation (NCE)

$$\text{loss}(w, v) = \log(1 + \exp(-in_w^T out_v)) + \sum_{\text{random } v'} \log(1 + \exp(in_w^T out_{v'})) \quad \text{[Mikolov et al., 2013]}$$

Представления word2vec (skipgram)

- Обучение на предсказании контекста по слову
 - Обучение на корпусе текстов без разметки
- Предсказываем каждое слово из контекста отдельно
 - Используются представления *in*
- Достоинства
 - Ближайшие соседи (cosine distance = норм. скал. произв., корпус GoogleNews)
 - Арифметика над представлениями
 - $\text{king} - \text{man} + \text{woman} = \text{queen}$



- Популярные представления – GloVe [Pennington et al., 2014]

Source: [Mikolov et al., 2013]

Представления fastText (skipgram)

Как в word2vec:

- Обучение на предсказании контекста по слову
 - Обучение на корпусе текстов без разметки
- Предсказываем каждое слово из контекста отдельно

Новая идея:

- Добавить информацию о символах слова (через n-граммы)

$$\text{in}_w = \text{word}_w + \sum_{p \in \text{n-grams}(w)} \text{part}_p \quad \text{in}_w^T \text{out}_v \Rightarrow \text{word}_w^T \text{out}_v + \sum_{p \in \text{n-grams}(w)} \text{part}_p^T \text{out}_v$$

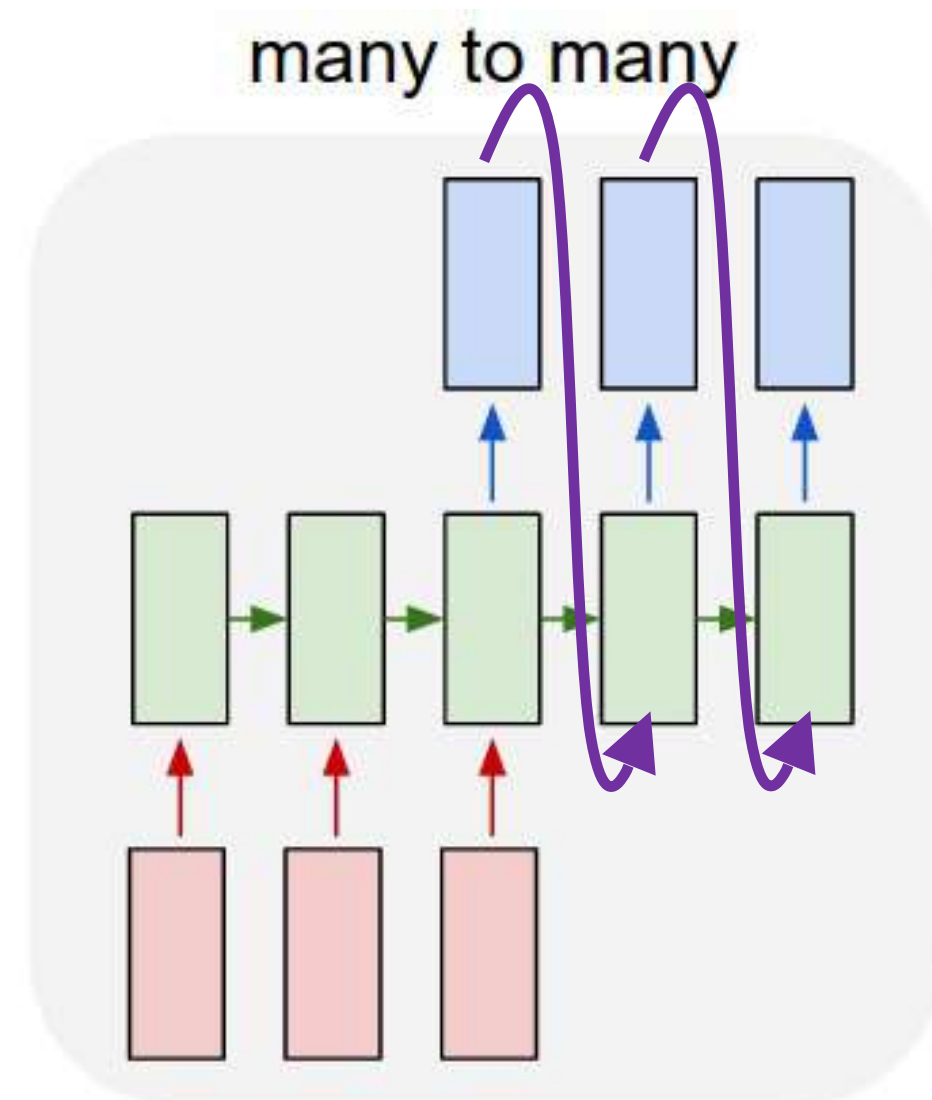
- “where” = “<where>”, “<wh”, “whe”, “her”, “ere”, “re>”
- Важно использовать длинные n-граммы ($n \leq 6$)
- Достоинства:
 - Близость по написанию
 - Слова вне словаря, опечатки и т.д.

**Код и данные на
fasttext.cc**

[Bojanowski et al., 2017]

Модель seq2seq

- Модели для предсказания последовательностей разной длины



Входы, память, выходы

Входы – представления входов

Память – слои RNN

Выходы – шансы слов из словаря
(logits, идут в logsoftmax)

Аutoreгрессионные связи (→) передают
решение о текущем слове


На входы → подаются представления
выходного алфавита

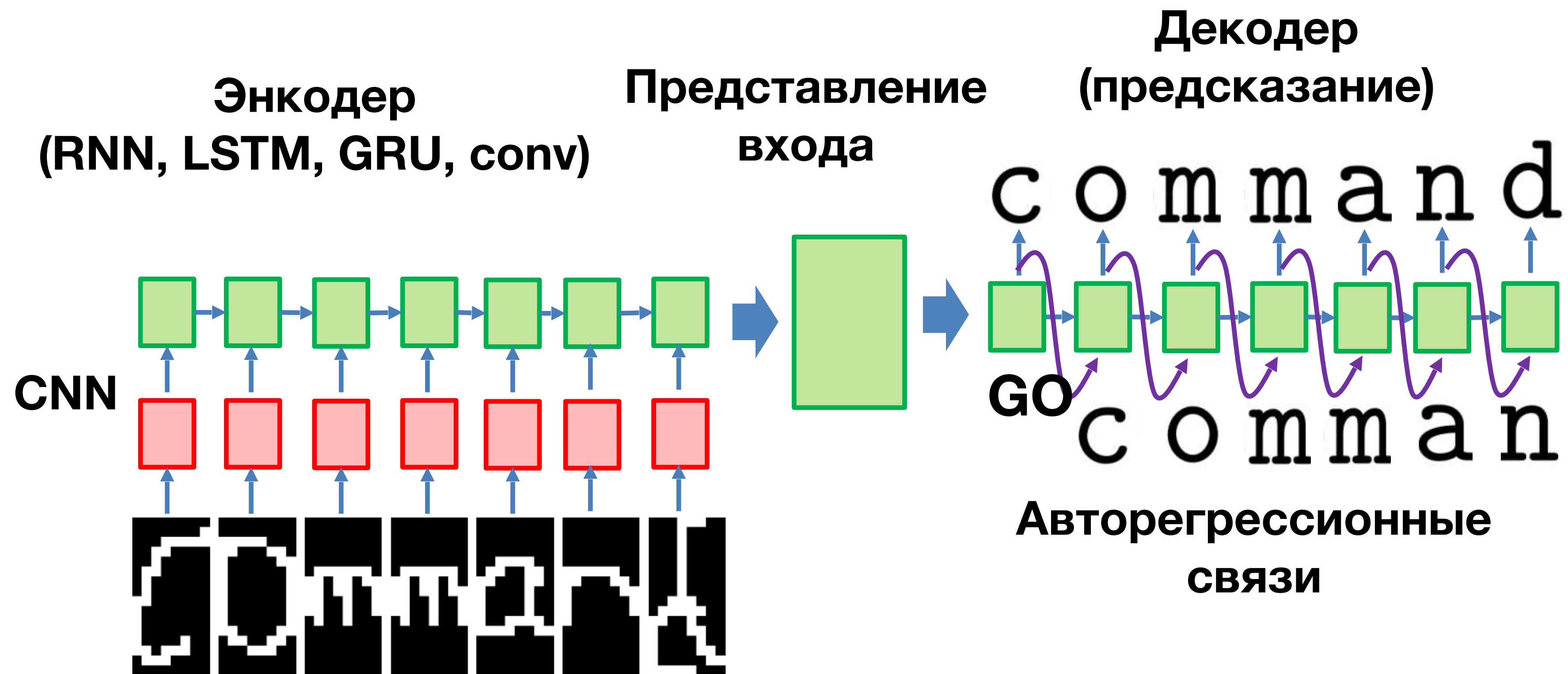
image credit: **Andrej
Karpathy**

[Sutskever et al. , 2014]

Последовательное предсказание

- Пример:

 → command



Если не фиксирована длина
выхода, то используют символ EOS

Обучение авторегрессивных моделей

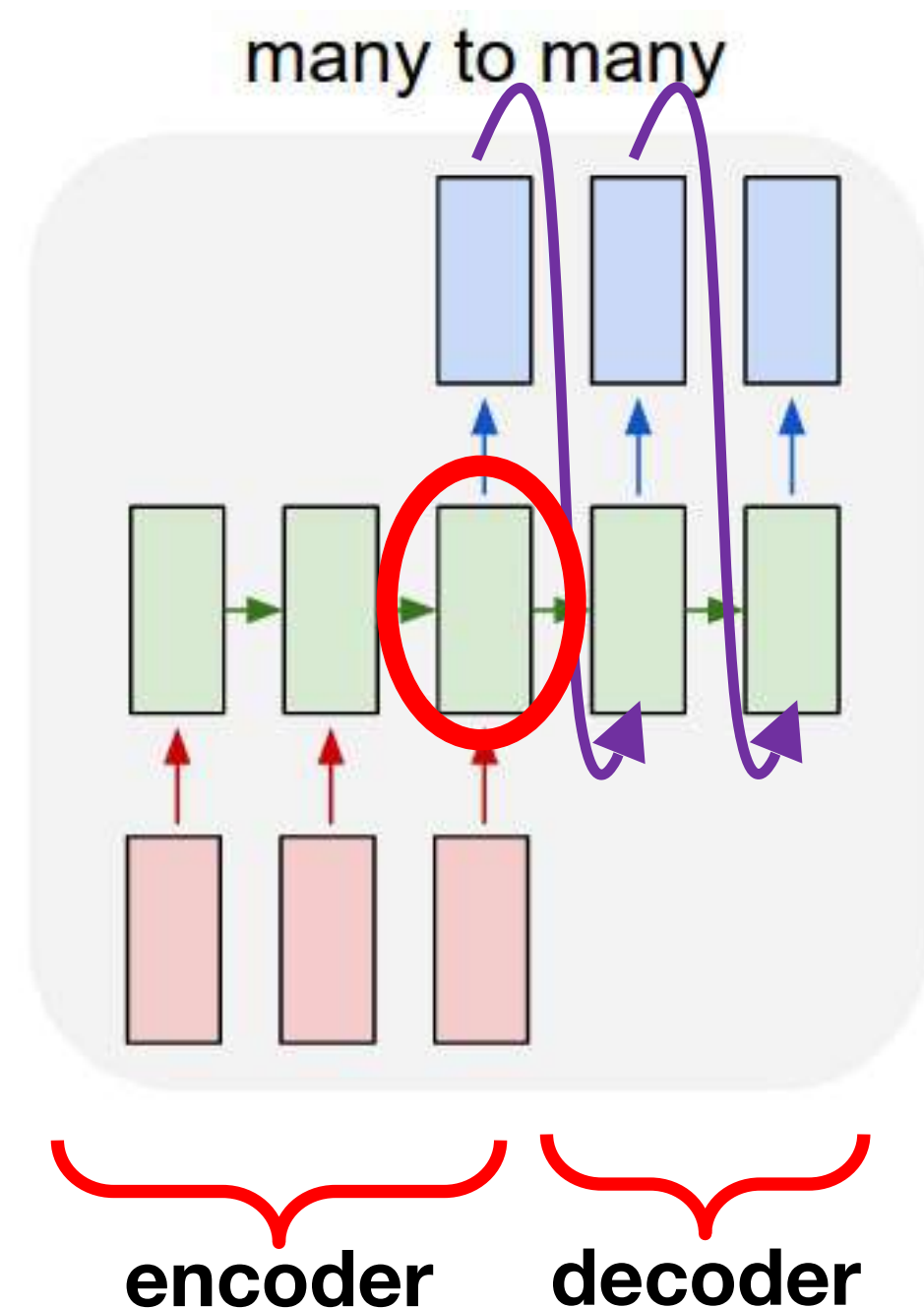
- Обычный способ – метод максимального правдоподобия
- На каждом шаге декодера – softmax и log-loss

$$P(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta}) = P(y_1 \mid \mathbf{x}, \boldsymbol{\theta})P(y_2 \mid y_1, \mathbf{x}, \boldsymbol{\theta})P(y_3 \mid y_2, y_1, \mathbf{x}, \boldsymbol{\theta}) \dots$$

- Teacher forcing – на вход декодеру подаются правильные ответы
- Проблема:
 - Модель видит только правильные траектории
 - Не знает, что делать при ошибке
 - Как исследовать траектории (exploration)?
 - Связь с обучением с подкреплением (RL)

Модель seq2seq

- Модели для предсказания последовательностей разной длины



Модель encoder-decoder

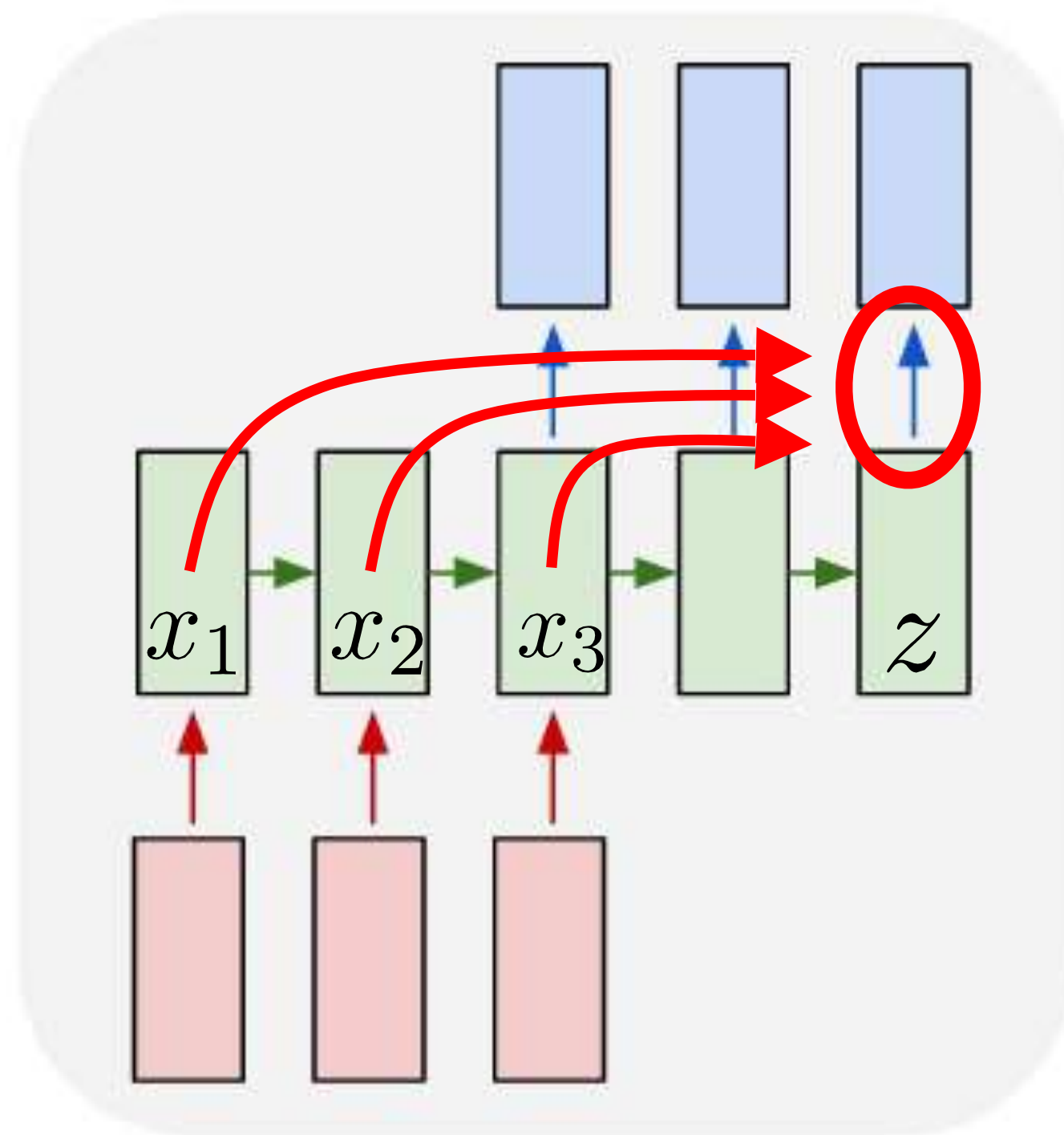
○ – представление всего входа

Модель плохо работает для длинных последовательностей

Причина: представление входа – вектор фиксированной размерности (не может представить весь текст)
Решение: механизм внимания (attention)

Модель seq2seq с вниманием

- Модели для предсказания последовательностей разной длины



Внимание «выбирает релевантные элементы памяти»

Модель внимания:

релевантность

$$s_i := \text{score}(x_i, z) = \begin{cases} x_i^T z \\ W[x_i; z] \end{cases} \quad \begin{matrix} \mathbf{W} - \text{параметры} \\ \text{модели} \end{matrix}$$

вероятности $a_1, a_2, \dots := \text{softmax}(s_1, s_2, \dots)$

контекст $c := \sum_i a_i x_i$

новые признаки $\tilde{z} := [c; z]$

soft-argmax

Transformer

- Статья – Attention Is All You Need

- Архитектура:

- Multi-head self attention

(Q = query, K = key, V = value, d_k = dimension = 64, $\kappa = 8$)

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O$$

$$\text{where head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

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

- 2-layer NN with ReLU

- Encoding: token and positional

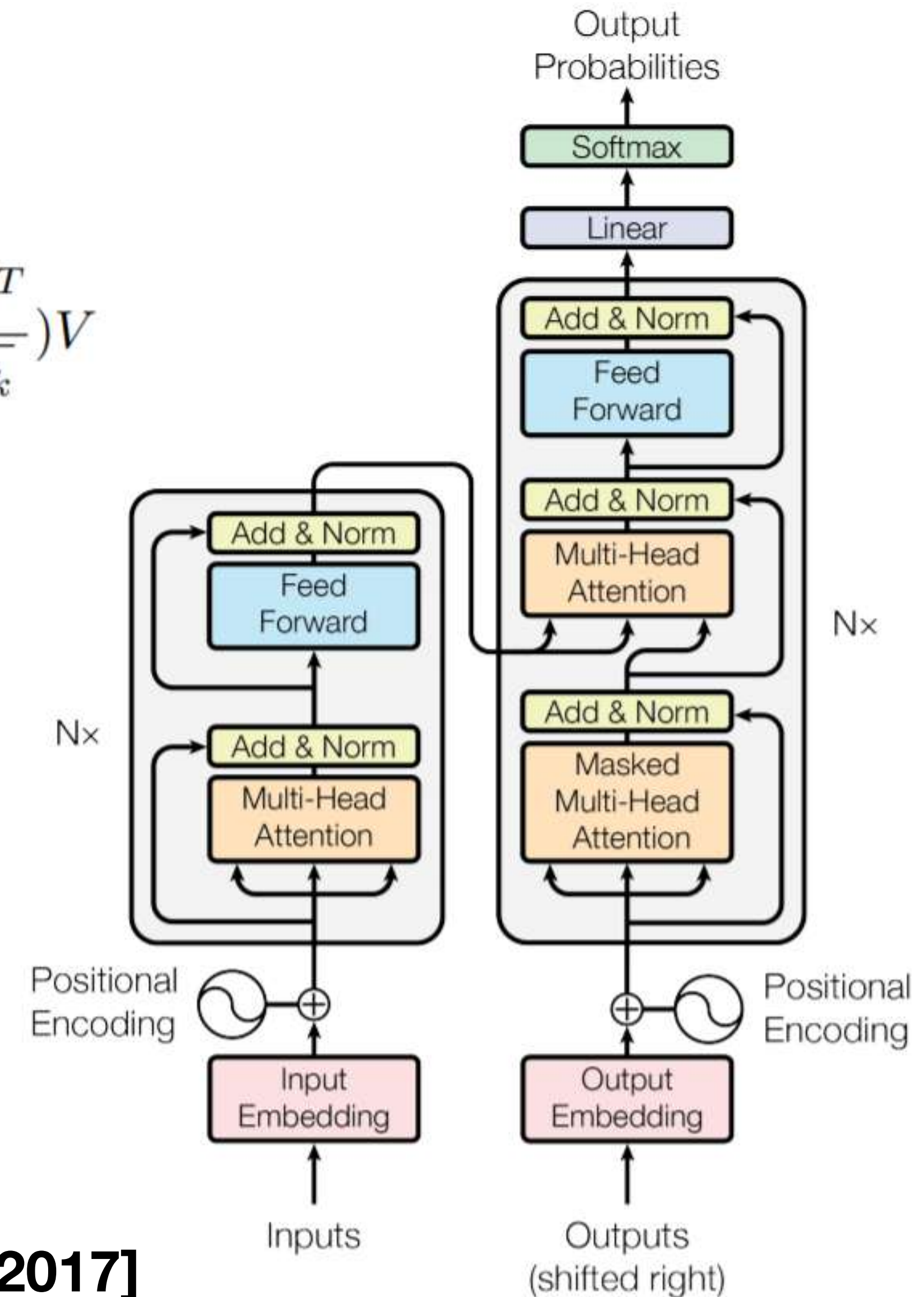
- Positional – sin и cos от длины, бывают обучаемые и не только

- В декодере – маска будущего!

- SOTA в переводе и др.!

- Поиск архитектуры [So et al., 2019]

- Ускорение: Reformer, Linformer, etc.



[Vaswani et al., 2017]

Словарь из Byte Pair Encoding (BPE)

- Размер словаря – важный параметр!
 - Большой => мало слов на редкие позиции, медленно, память
 - Маленький => много слов вне словаря
- Построение словаря BPE - итеративно
 - Инициализация – из токенов символов (unicode - осторожно)
 - Итеративное склеивание самых частых пар
 - Пересечение границ слов?
 - Символы типа знаков препинания?
- Позволяет делать представления любого слова
- Размер словаря – контролируемый параметр

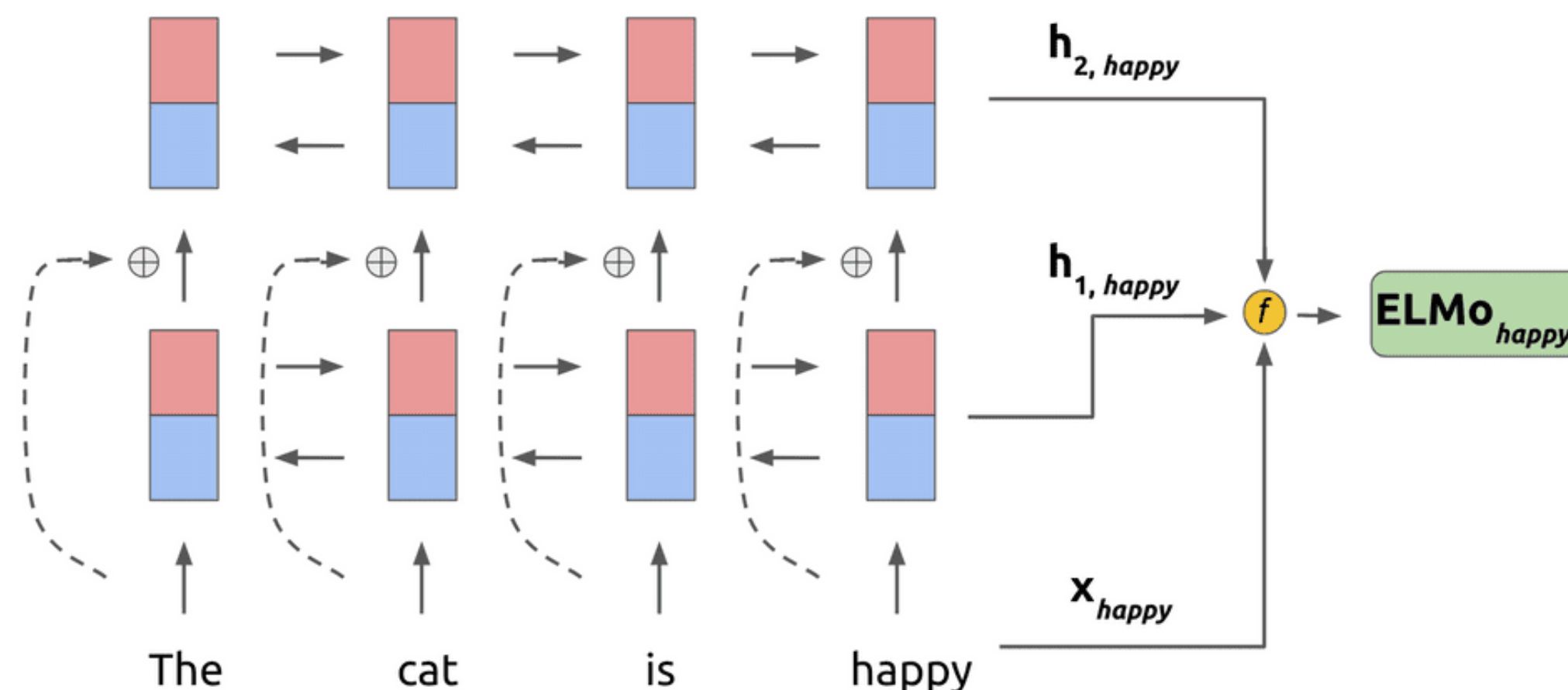
Предобученные представления слов из языковых моделей

ELMo

- ELMo = Embeddings from Language Models
- Контекстно зависимые представления!
- Архитектура:
 - Представления слов = свертки поверх представлений символов
 - Легко обрабатывать слова вне словаря
 - 2 модели поверх – forward и backward (2-layer LSTM + skip con.)
 - Итоговое представление – линейная комбинация представлений слоев!

[Peters et al., 2018; AllenNLP]
github.com/allenai/allennlp

- Обучение:
 - Forward – след. слово
 - Backward – пред. слово



BERT

- BERT = Bidirectional Encoder Representations from Transformers
- Очень большой трансформер:
 - L – глубина, H – размерность внутри, A – число голов multi-head attention
 - BERT_{BASE}: L=12, H=768, A=12, Total Parameters=110M
 - BERT_{LARGE}: L=24, H=1024, A=16, Total Parameters=340M

- Обучение:
 - Masked LM: 15% tokens selected randomly
 - 80% - [MASK]
 - 10% - исходное слово
 - 10% - случайное слово
 - Next sentence

Input = [CLS] the man went to [MASK] store [SEP]
he bought a gallon [MASK] milk [SEP]

Label = IsNext

Input = [CLS] the man [MASK] to the store [SEP]
penguin [MASK] are flight ##less birds [SEP]

Label = NotNext

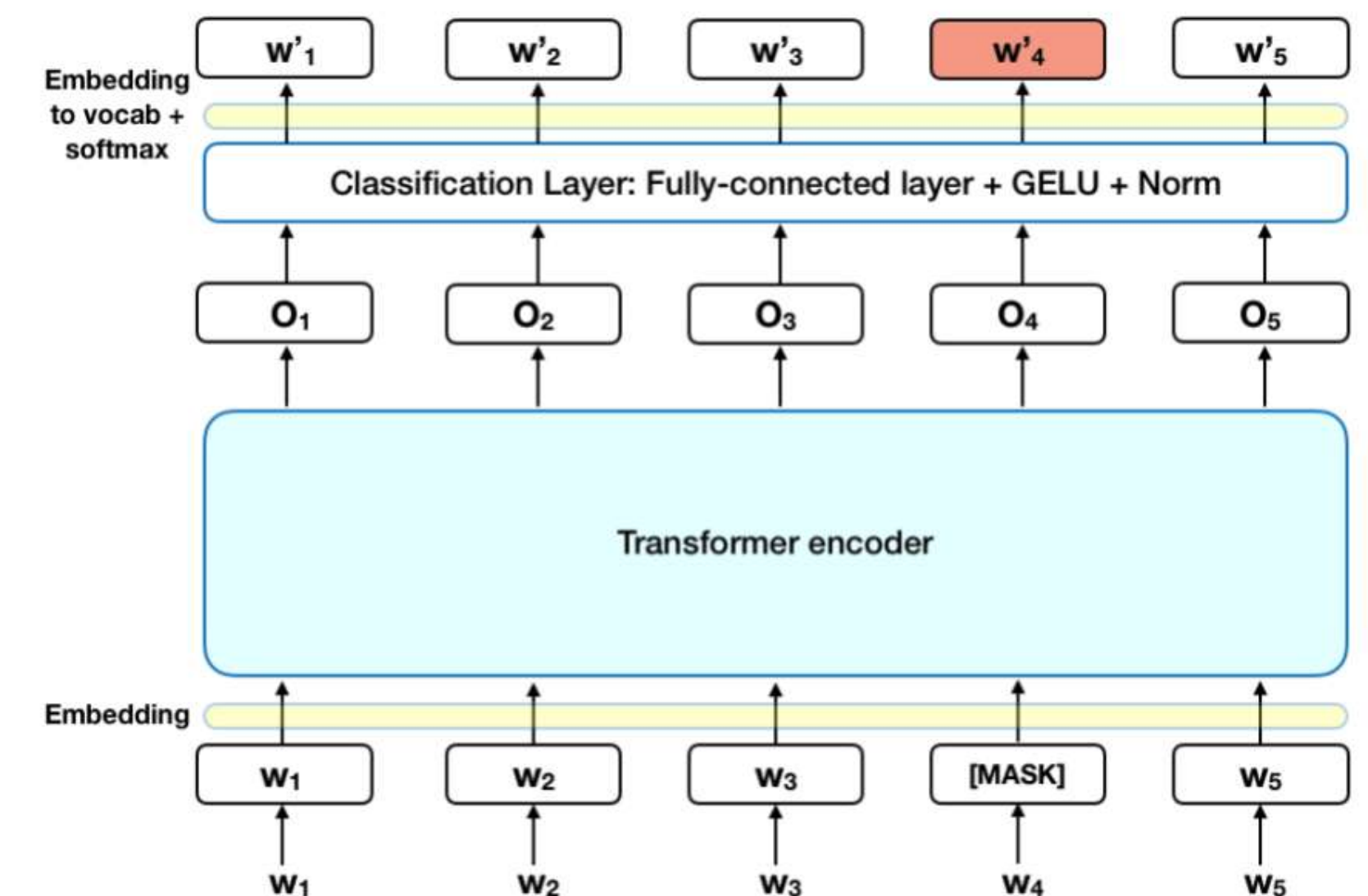


image credit: Rani Horev

[Devlin et al., 2018]

BERT friends: RoBERTa, ALBERT, T5, GPT-3

- Тренд: увеличение моделей и данных
- Обучение очень сложное и дорогое
- Высокая чувствительность к гиперпараметрам
- RoBERTa: BERT был существенно недообучен
 - Можно стать SOTA изменив параметры и обучая дольше
 - Большой батч, byte-level (а не character-level, unicode!) BPE, динамические маски для предложений
- Можно (часто нужно!) использовать готовые модели!
- GPT-3 – 175B параметров, обучение стоит миллионы \$
 - Может работать для предсказания во few-shot режиме без fine-tuning
- Отличная библиотека:
 - github.com/huggingface/transformers

[Liu et al., 2019; Facebook]

[Lan et al., 2019; Google]

[Raffel et al., 2019; Google]

[Brown et al., 2020; OpenAI]

Заключение

- Обработка языка активно использует нейросети
- Очень большая область – много задач
 - Есть успехи, причем множество
- Представления, Seq2seq, внимание, transformer, BERT, etc.
- Понимание смысла – очень сложная задача!