ML в NLP 101

Екатерина Владимировна Еникеева

20 сентября 2022

Автоматическая обработка естественного языка, лекция 3

План лекции

- Основные задачи машинного обучения
- Простейший классификатор – наивный байес
- Более специфические: sequence tagging, seq2seq, …
- Линейные модели ML
- Нейросети: ffnn, rnn, encoder-decoder, bert
- Как получаются эмбеддинги ?

Базовая задача ML

Обучающие данные:

- Вход (*input*): векторное представление (*feature representation*) *x* – текста отзыва, конкретного слова …
- Выход (*output*) *y* – класс (label), число, вероятность

Цель:

- Построить модель (алгоритм+набор параметров), чтобы хорошо предсказывать *y* по данным *x*

Представление входных данных

- текст > предложения > токены
- *bag-of-words* (мешок слов)
- *one-hot encoding*: словарь V > сопоставляем каждому слову вектор ($x_0 \ldots x_{|V|}$), где $x_i=1$, $x_k=0 : k \neq i$
- для представления текста — складываем *one-hot* всех слов > вектор частот слов из словаря в документе (term frequency – *tf*)

Feature extraction

- Как представить текст?
  - Bag of words
  - Некоторые слова важнее других -> tf*idf, словари ключевых слов …
  - Количественные характеристики: длина текста, длина предложения … , глубина дерева …
- Как представить слово?
  - Отдельные символы – не все одинаково важны
  - Капитализация, суффикс, префикс
  - Контекст: слова, теги, суффиксы …

Feature learning

Embeddings / distributed representations

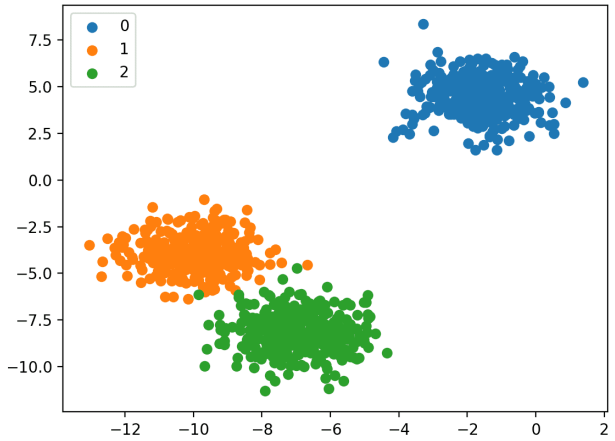
- Pre-trained embeddings / предобученные представления слов (word2vec etc.) — обычно обучаются на задаче LM
  - могут учитывать char-level признаки
- Скрытые представления для конкретной задачи

Представление *y*

- дискретные метки / классы
  - бинарная классификация (спам – не спам…)
  - многоклассовая классификация (POS-теги)
  - multi-label классификация (хэштеги; один текст – может быть несколько хэштегов)
- непрерывные значения – зарплата по описанию вакансии / цена продукта по тексту …
  - вероятность каждого слова в словаре

Типы задач

- Классификация:
  - Бинарная: $f(x) \rightarrow y; y \in \{0, 1\}$
  - Многоклассовая (мультиклассификация): $y \in \{c_0, \dots c_n\}$, где n – число класов



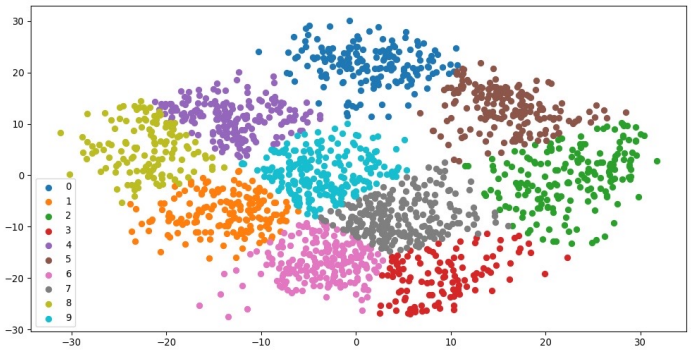
Частный случай классификации – *разметка последовательности / sequence labeling*

Типы задач

- Регрессия:

  $f(x) \to y$, где y - число

- Кластеризация



Классификаторы

- Генеративные / generative

  моделируем генерацию классами входных данных

  $$p(y, x)$$

- Дискриминативные / discriminative

  учимся разделять классы по входным признакам

  $$p(y|x)$$

Naïve Bayes

Наивный Байесовский классификатор:
пусть есть K признаков

$$p(y,x) = p(y)p(x|y) = \ p(y) * \prod_{k=1}^{K} p(x_k|y)$$

Как оценить вероятности через частоты:

$$P(x_i|c_j) = \frac{count(x_i, c_j)}{\sum_{x_k \in X} count(x_k, c_j)}$$

Naïve Bayes

	Cat	Documents
Training	-	just plain boring
	-	entirely predictable and lacks energy
	-	no surprises and very few laughs
	+	very powerful
	+	the most fun film of the summer
Test	?	predictable with no fun

prior probabilities $P(-) = \frac{3}{5} \quad P(+) = \frac{2}{5}$

likelihoods

$$P(\text{“predictable”}|-) = \frac{1+1}{14+20} \quad P(\text{“predictable”}|+) = \frac{0+1}{9+20}$$

$$P(\text{“no”}|-) = \frac{1+1}{14+20} \quad P(\text{“no”}|+) = \frac{0+1}{9+20}$$

$$P(\text{“fun”}|-) = \frac{0+1}{14+20} \quad P(\text{“fun”}|+) = \frac{1+1}{9+20}$$

$$P(-)P(S|-) = \frac{3}{5} \times \frac{2 \times 2 \times 1}{34^3} = 6.1 \times 10^{-5}$$

$$P(+)P(S|+) = \frac{2}{5} \times \frac{1 \times 1 \times 2}{29^3} = 3.2 \times 10^{-5}$$

Задачи классификации

Задача	Объекты	Классы
Language Modelling	контекст	слова в словаре
POS-tagging	словоформа / токен	POS-теги
Морфоанализ	словоформа / токен	целый тег из набора граммем; каждый признак – отдельная задача классификации
Parsing	слова в предложении	связи

Функция потерь

loss function

- обозначим предсказанное $\hat{y}$
- y – числа: $loss = y - \hat{y}$ (абсолютная ошибка)

$$MSE = \frac{1}{n}\sum_{i=0}^{n}(y - \hat{y})^2$$

Используется при подборе параметров: на каждом шаге стараемся минимизировать ошибку

Cross-entropy

- для классификации нам важно понимать, предсказали ли мы нужный класс

Если $y \in (0, 1)$, то для конкретного примера:

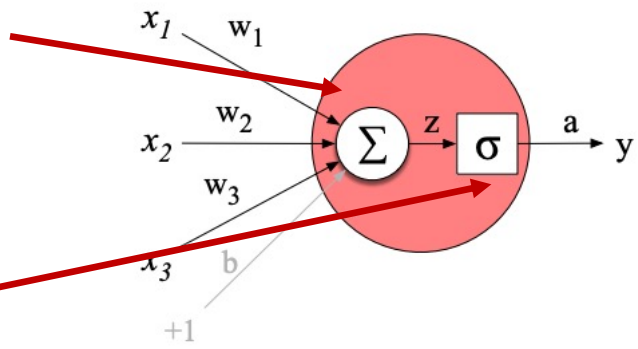
$$L_{CE}(y, \hat{y}) = -\log p(y|x) = -[y \log \hat{y} + (1 - y) \log(1 - \hat{y})]$$

Нейрон (computational unit)

Нейрон / перцептрон (perceptron)

Взвешенная сумма входов + сдвиг (*bias term*)

$$z = wx + b = b + \sum_{i=0}^{n} w_i x_i$$



Функция активации

$$y = f(z) = \begin{cases} \sigma(z) = \dfrac{1}{1+e^{-z}} & - \ sigmoid \\ \tanh z = \dfrac{e^{z} - e^{-z}}{e^{z} + e^{-z}} & - \text{гиперболический тангенс} \\ ReLU(z) = \max(z, 0) & - \ rectified \ linear \ unit \end{cases}$$

Feed-forward neural network

Полносвязная (fully-connected) *сеть прямого распространения*

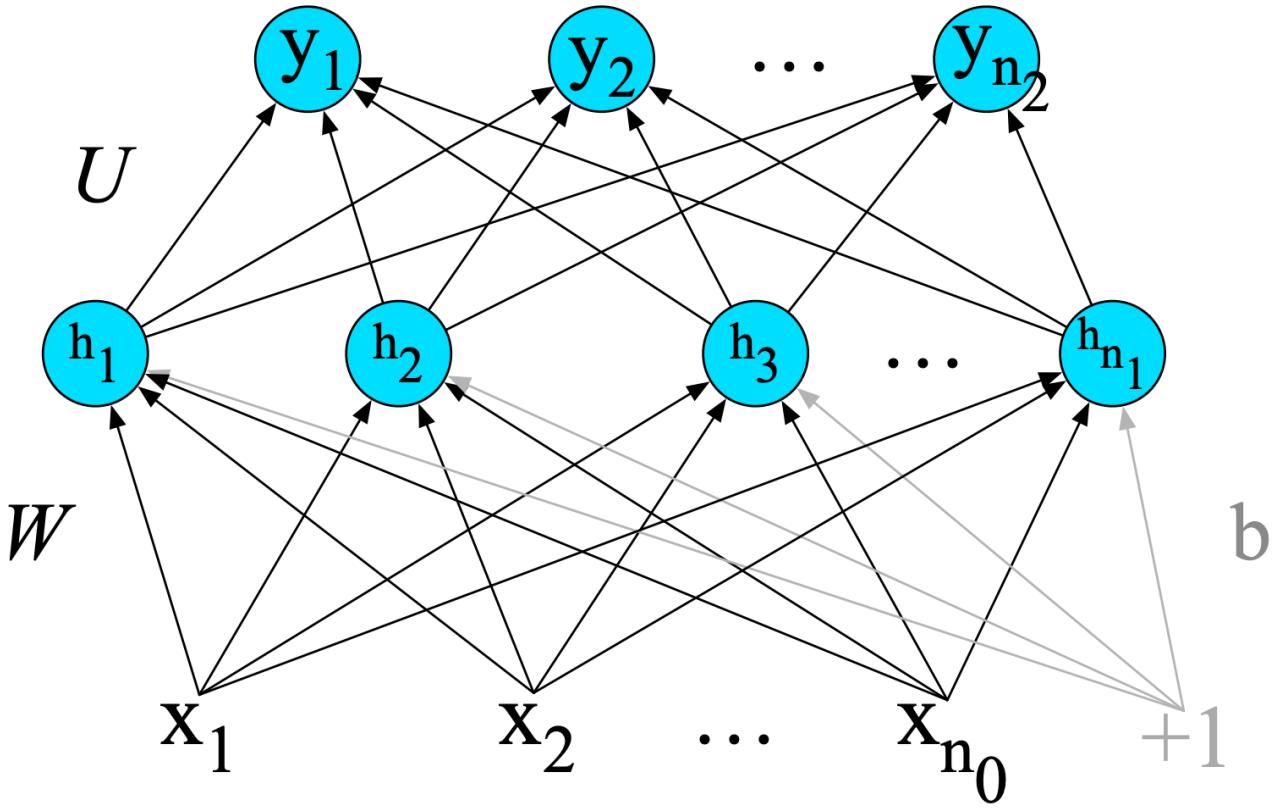
- Скрытый слой, матрица весов W

$$h = \sigma(Wx + b)$$

- Выходной слой, матрица весов U

$$z = Uh$$
$$\hat{y} = softmax(z)$$

Feed-forward neural network



Функция потерь

negative log likelihood loss:

$$L_{CE}(y, \hat{y}) = -\sum_{i=1}^{C} y_i \log \hat{y}_i = -\log \hat{y}_i$$

Представляем y в обучающих данных как *one-hot vector*

Градиентный спуск

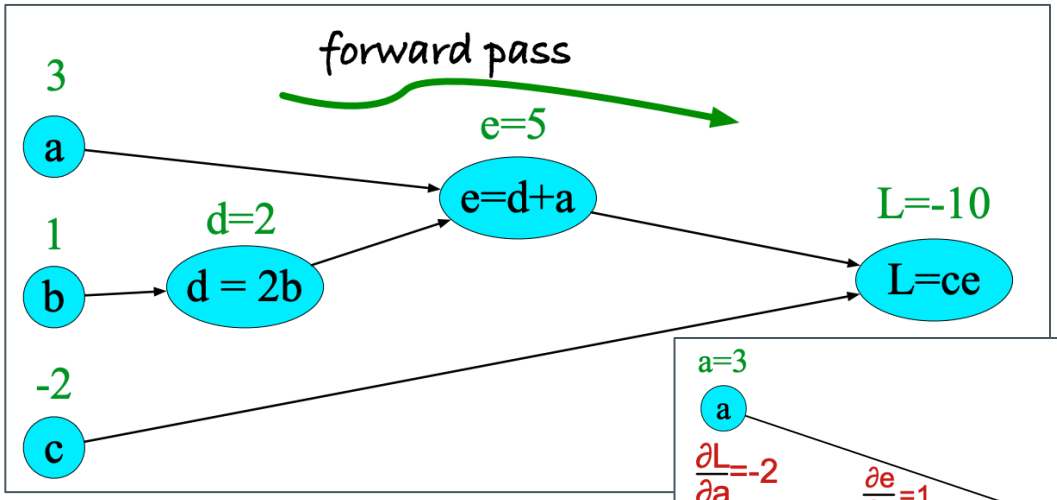
- Нужно распределить влияние ошибки по нескольким слоям

Error backpropagation — алгоритм обратного распространения ошибки
(*обратное дифференцирование — reverse differentiation*)

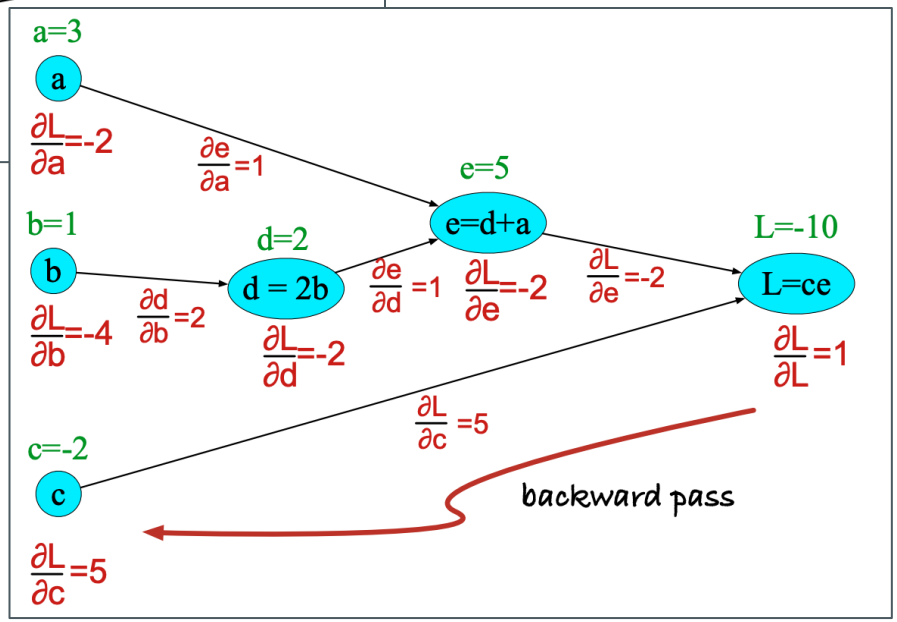
- Строим граф вычислений (computation graph) и пользуемся цепным правилом, которое дает производную композиции функций:

$$f(x) = u\big(v(x)\big) \Rightarrow \frac{df}{dx} = \frac{du}{dv} \cdot \frac{dv}{dx}$$

Error backpropagation



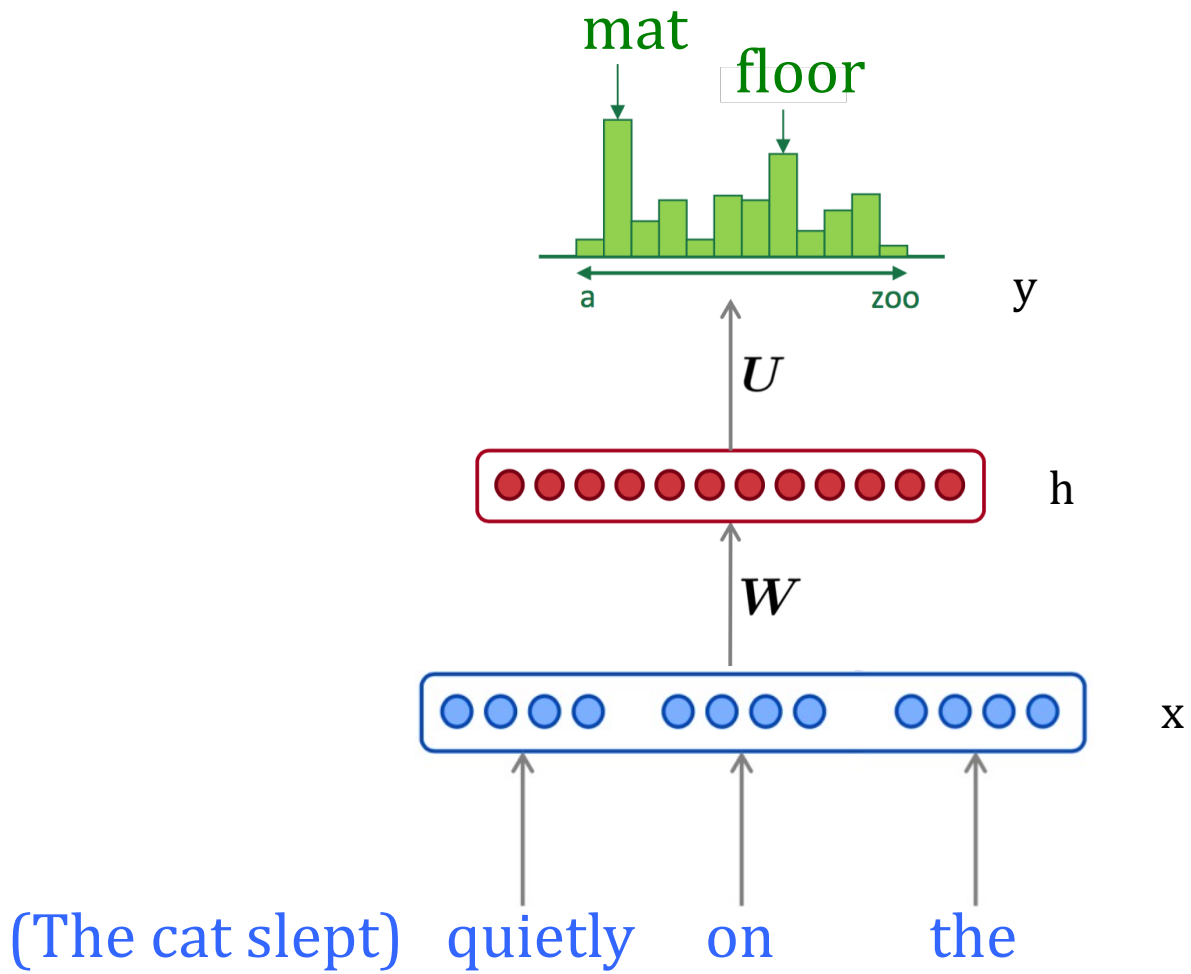
$$L(a, b, c) = c(a + 2b)$$



Входной слой

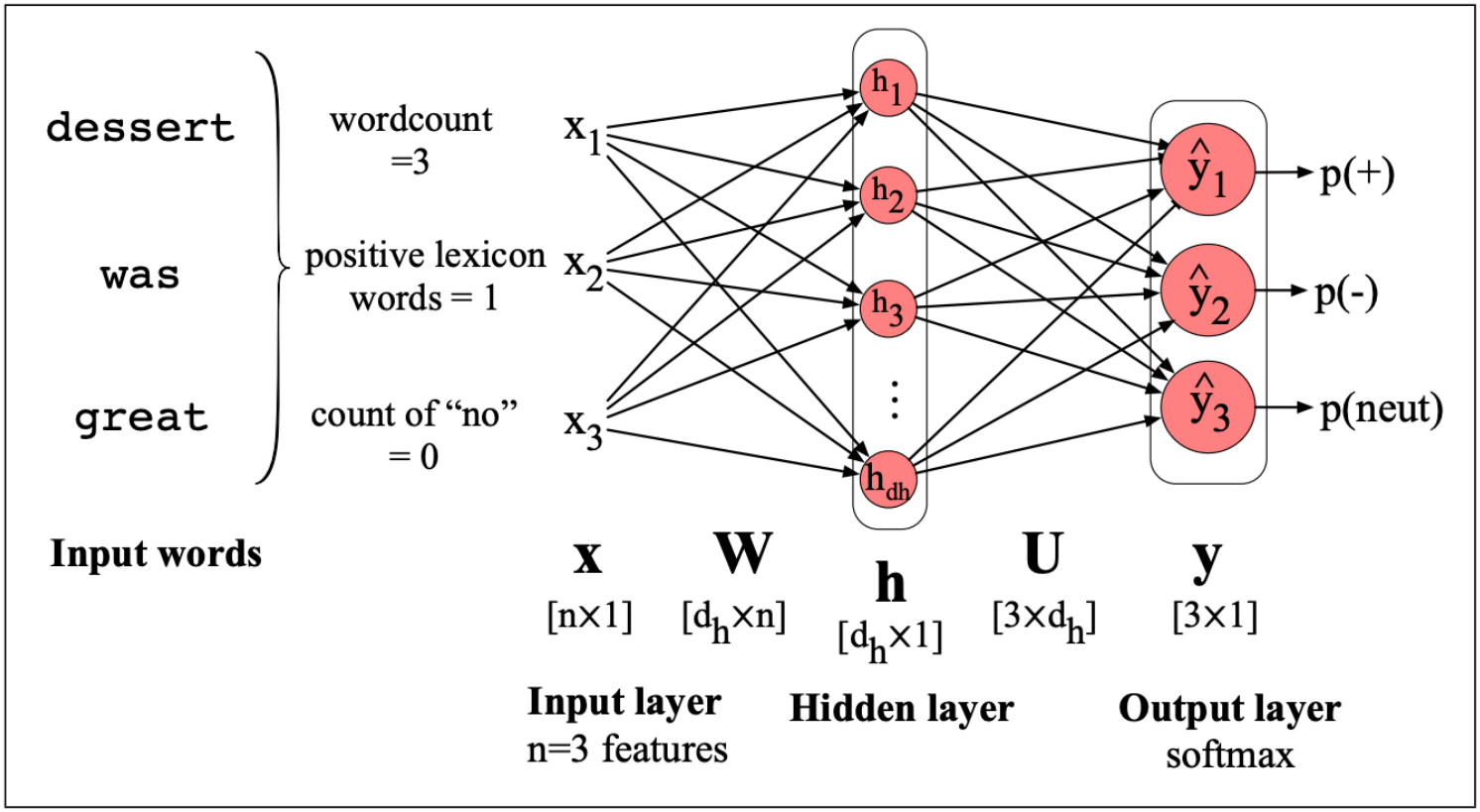
- Предобученные векторные представления отдельных слов (*pretrained embeddings*)
- *One-hot* вектора –> тогда выучиваем эмбеддинги в процессе – добавляем специальный слой – *projection/embedding layer*

Нейронные языковые модели



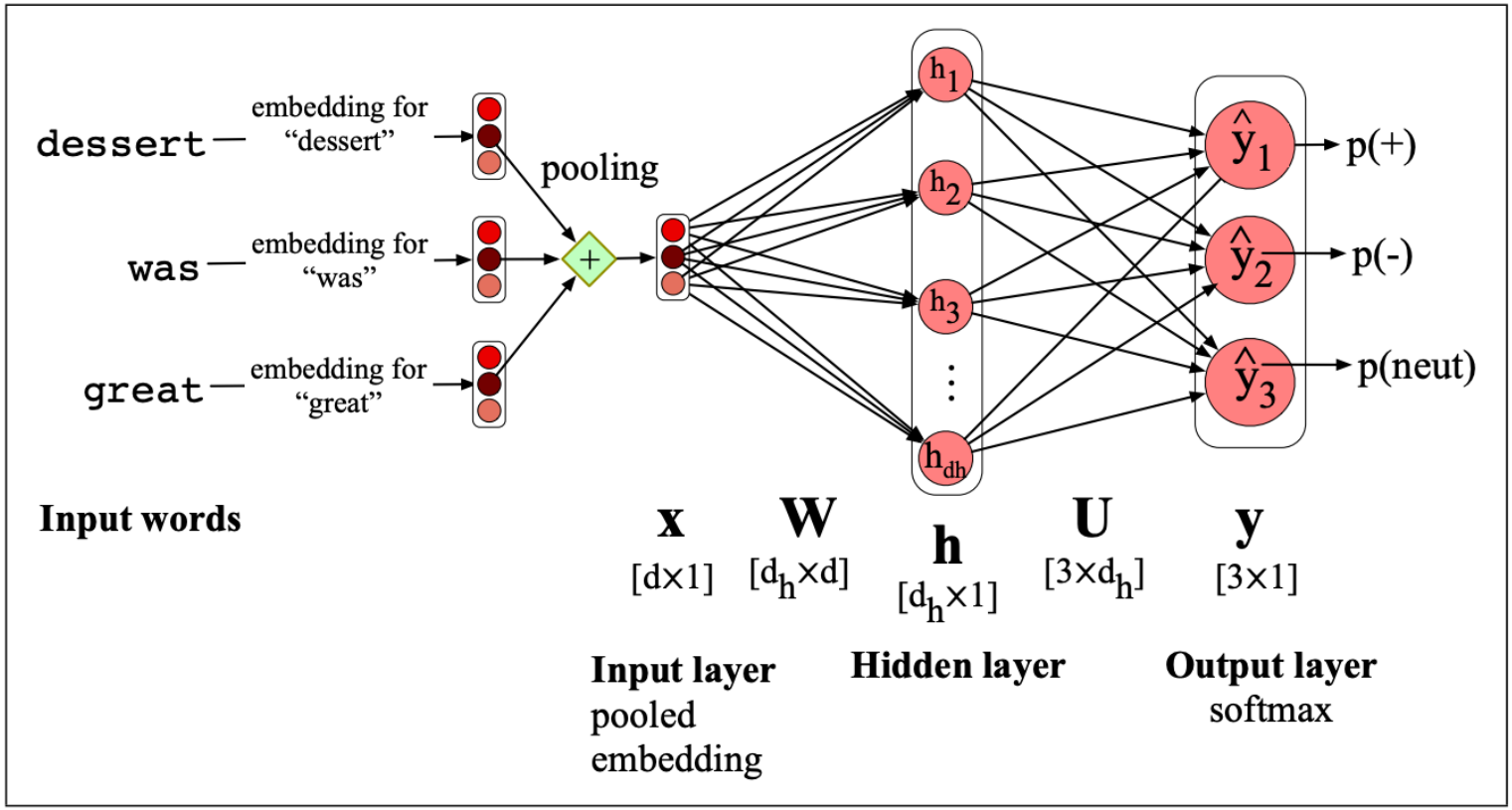
Sentiment classification

Feature-based approach:

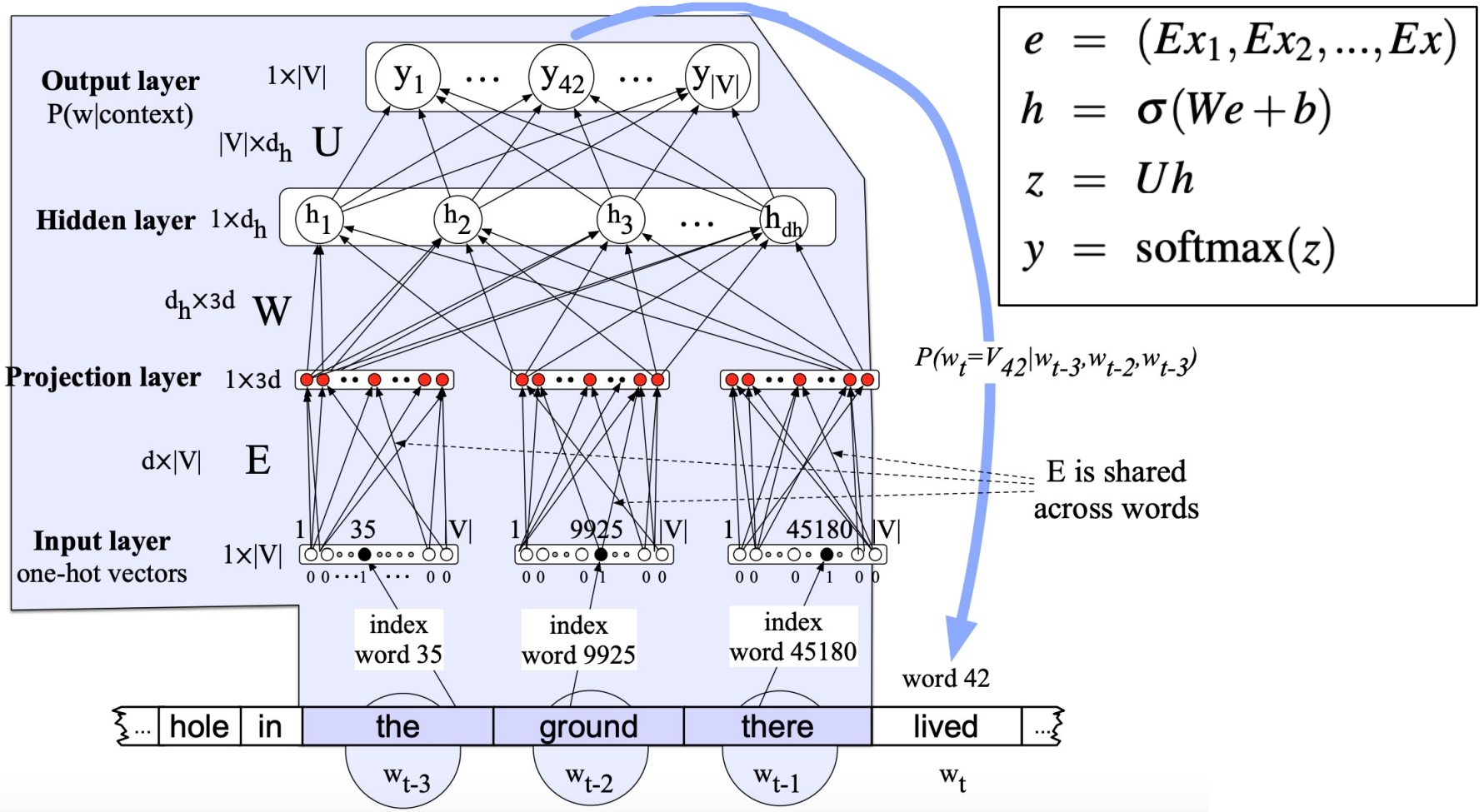


Sentiment classification

Embedding-based approach:



Обучение эмбеддингов



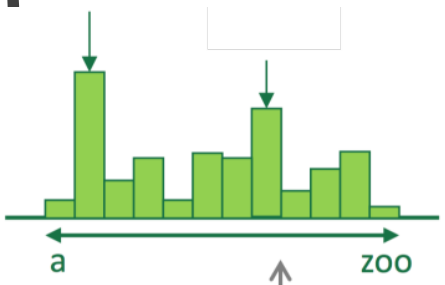
$$
\begin{aligned}
e &= (Ex_1, Ex_2, \ldots, Ex) \\
h &= \sigma(We + b) \\
z &= Uh \\
y &= \text{softmax}(z)
\end{aligned}
$$

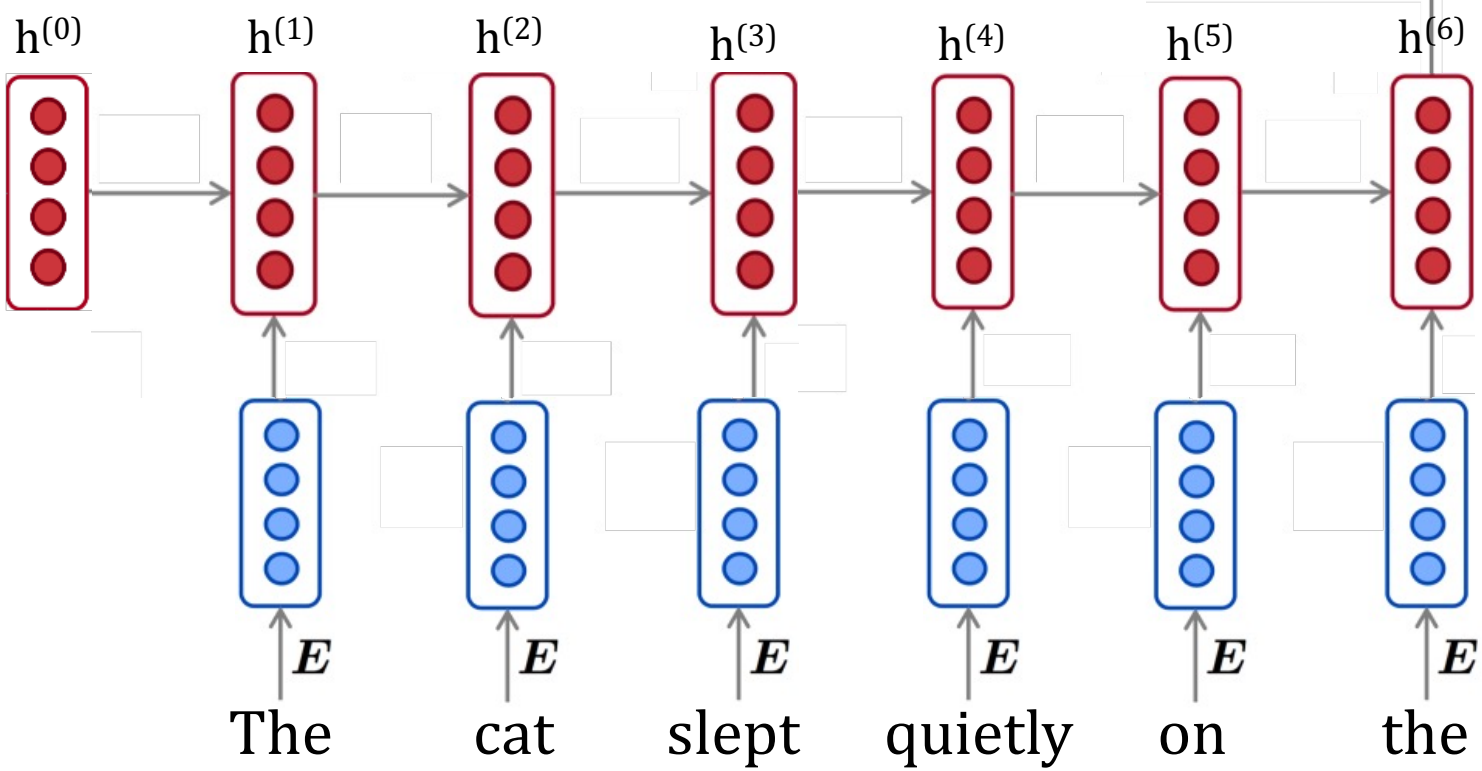
Недостатки FF-LM

- Ограниченный контекст:
  - Не выучиваются системные паттерны, которые оказываются на границах контекстного окна;
  - Не учитываются дистантные зависимости.

- Ещё?

RNaN-LM (1)





RNN-LM (2)

- Каждое слово превращаем в вектор e_{w_t}
- Скрытый слой

$$h^{(t)} = \sigma(W_h h^{(t-1)} + W_e e_{w_{t-1}} + b_1)$$

- Выходное распределение

$$\hat{y}_t = soft\max(Uh^{(t)} + b_2)$$

Улучшения RNN-LM

- Снова дистантные зависимости – проблема затухающего градиента (*vanishing gradient*)

> LSTM – Long Short Term Memory

> GRU – Gated Recurrent Unit

> Self-Attention (> Transformer)

RNN Sequence Labeling

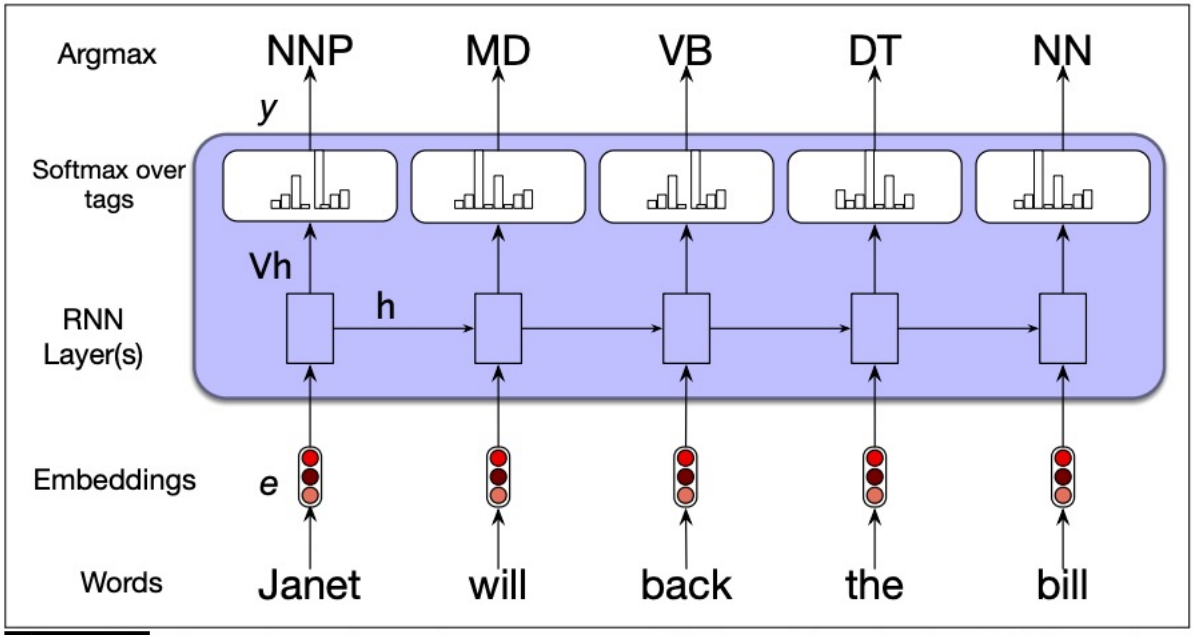
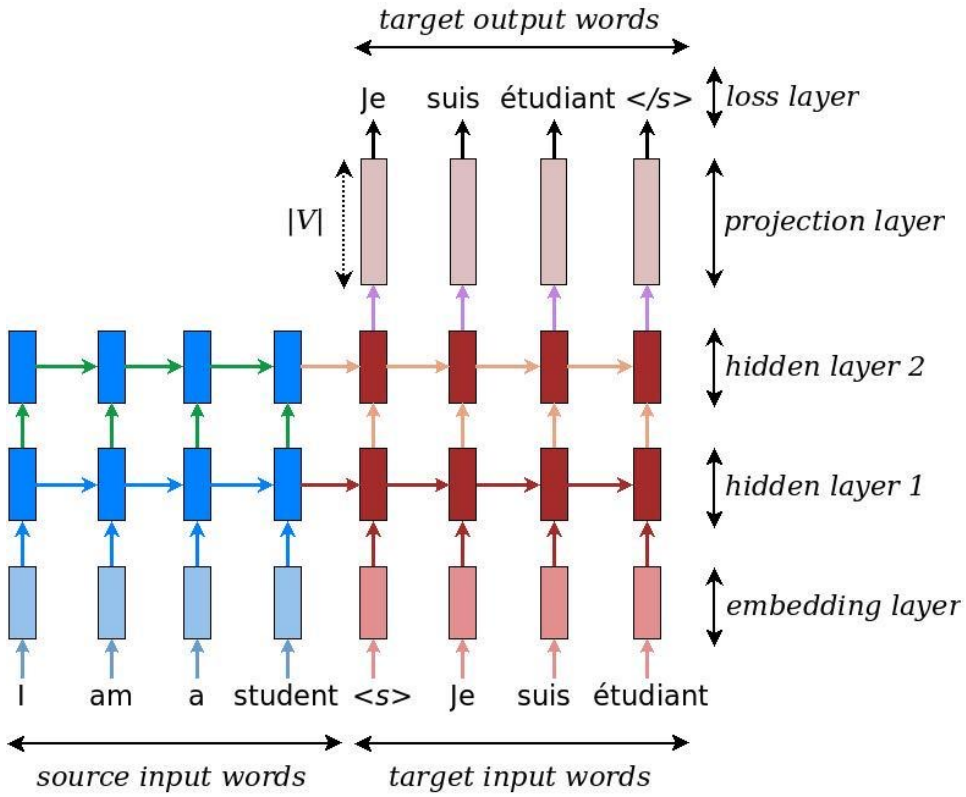


Figure 9.7 Part-of-speech tagging as sequence labeling with a simple RNN. Pre-trained word embeddings serve as inputs and a softmax layer provides a probability distribution over the part-of-speech tags as output at each time step.

Encoder-Decoder model



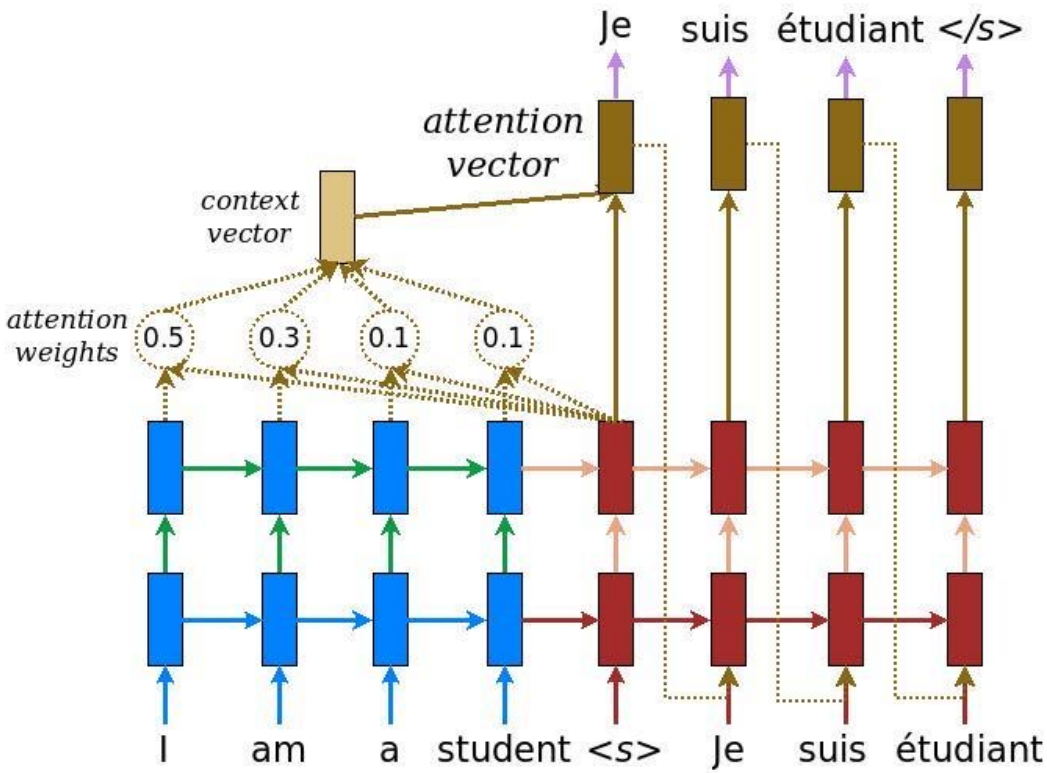
Идея из области машинного перевода:

- encoder: кодирует входной текст
- decoder: декодирует скрытое представление в перевод

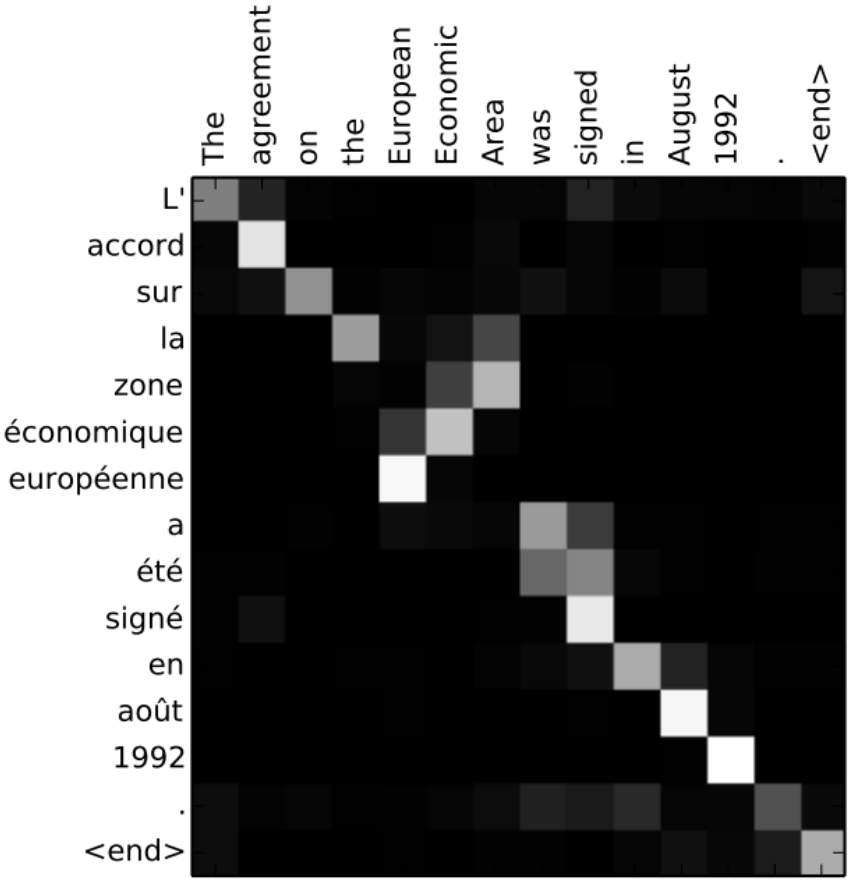
Attention is all you need

- Идея: давайте хранить скрытые представления для всех входных слов и пытаться учитывать их на каждом шаге декодера
- Теперь входное предложение – матрица **H**
- Вводим новый параметр – вектор α, который при умножении на матрицу **H** будет давать «вектор контекста» $\boldsymbol{c}$
- Вектор α будет показывать, насколько нужно учитывать каждое входное слово; получаем его с помощью состояния декодера $\boldsymbol{h^d}$ (например, скалярное произведение)

Attention


Je
suis
étudiant </s>
attention vector
context vector
attention weights
0.5
0.3
0.1
0.1
I
am
a
student </s>
Je
suis
étudiant

Интерпретация внимания в MT



Self-Attention

Вместо рекуррентных связей можем использовать контекст любого размера

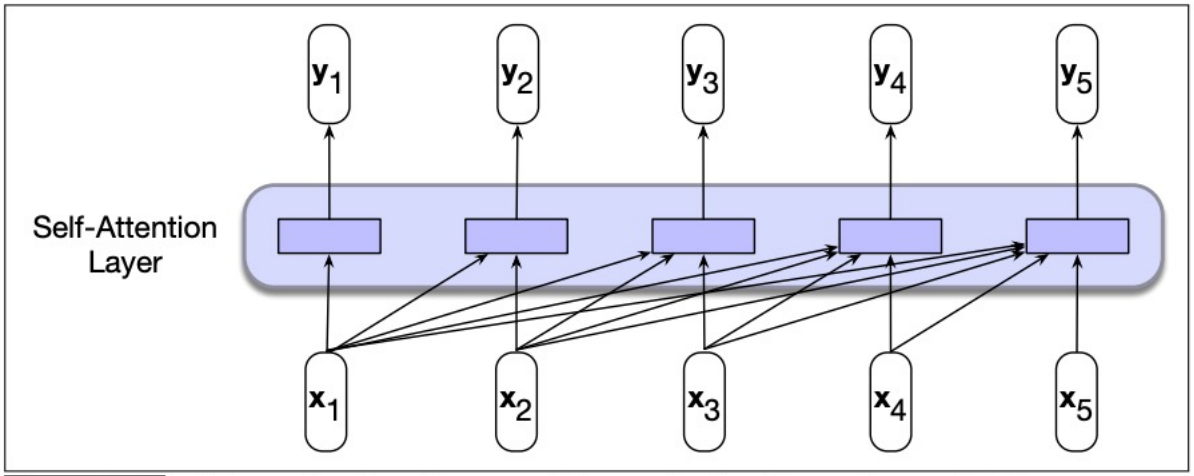


Figure 9.15 Information flow in a causal (or masked) self-attention model. In processing each element of the sequence, the model attends to all the inputs up to, and including, the current one. Unlike RNNs, the computations at each time step are independent of all the other steps and therefore can be performed in parallel.

Self-Attention

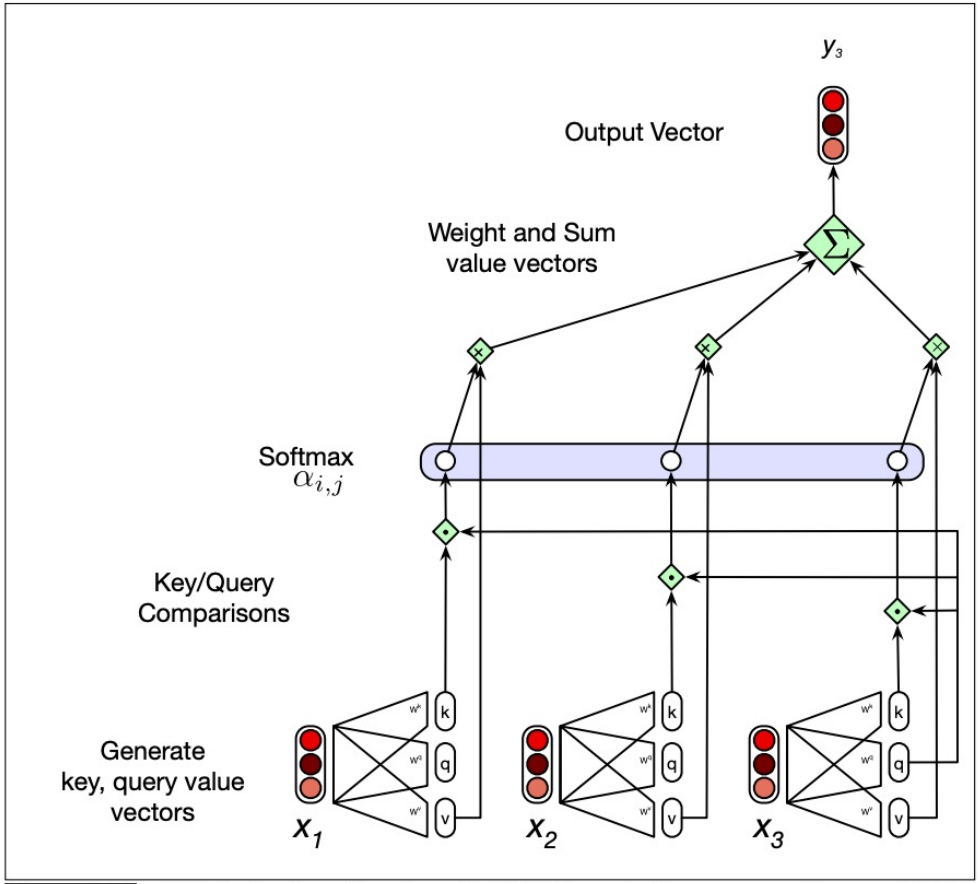


Figure 9.16 Calculating the value of $\mathbf{y}_3$, the third element of a sequence using causal (left-to-right) self-attention.

Трансформер

- **Self-Attention**
- **Multi-Head Attention** – несколько слоев self-attention, каждый со своими матрицами весов; затем объединяем их в один вектор с помощью ещё одной матрицы весов
- **Positional Encoding** — учитываем порядок

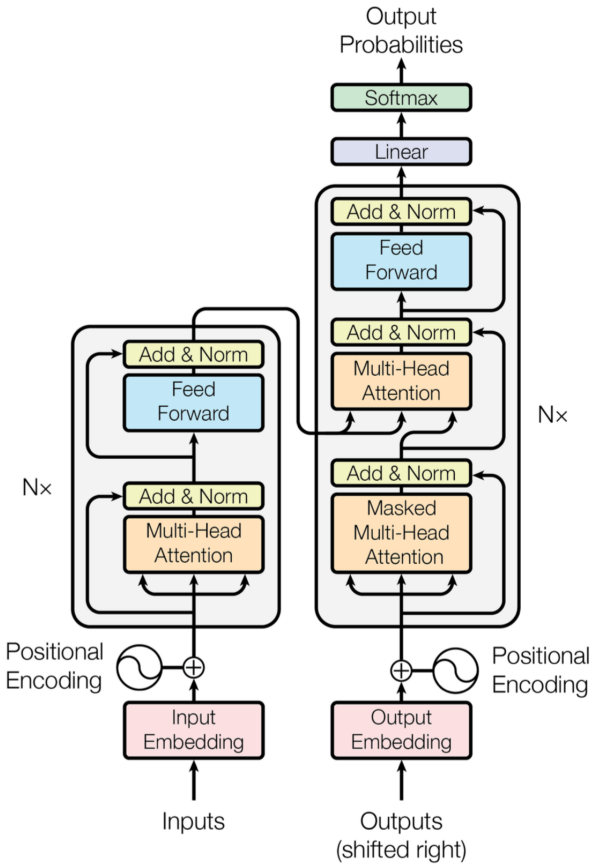
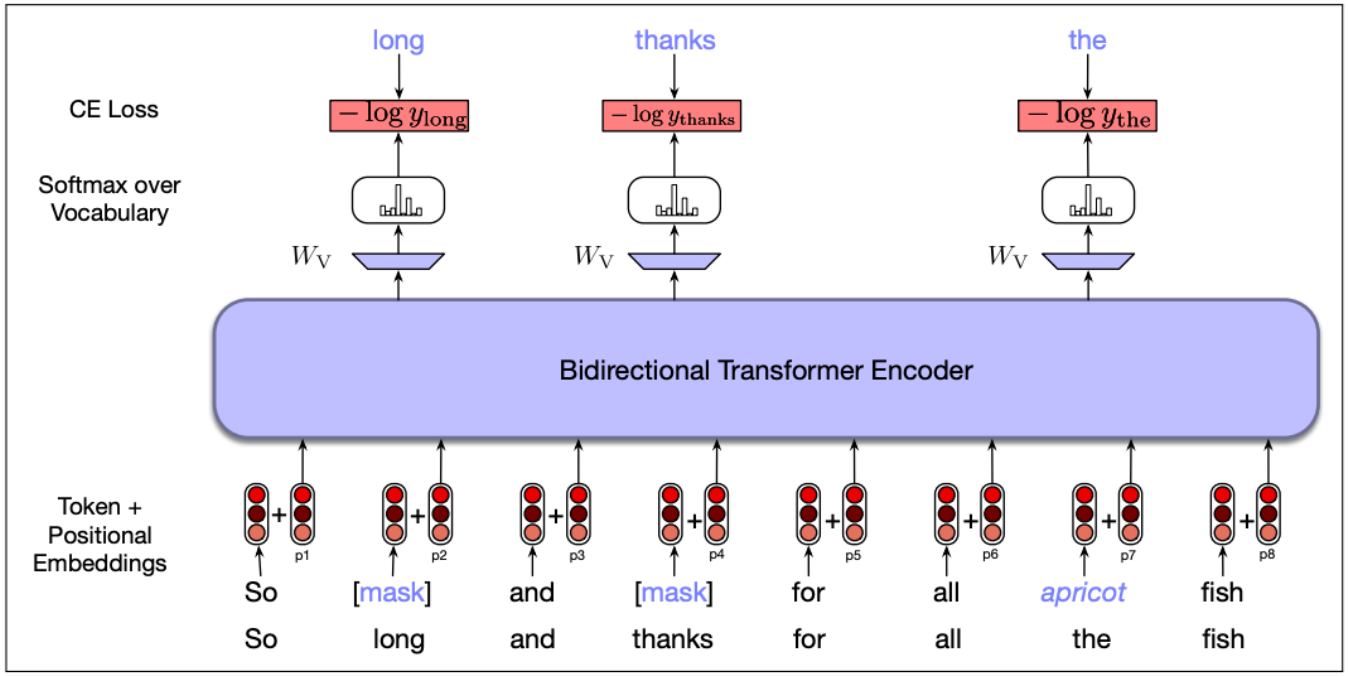


Figure 1: The Transformer - model architecture.

BERT=Bidirectional Transformer Encoders

Обучается задаче Masked Language Modeling (MLM):
Please turn ____ homework in.



Transfer Learning

- Pre-training
  - pretrained language models: обучаемся простой задаче, не требующей разметки, на больших данных
- Fine-tuning
  - используем параметры предобученной модели, чтобы дообучиться на других данных (например, более сложная разметка)
- Few-shot learning
  - GPT-3 etc.

Спасибо!

Вопросы?