

NLP Crash course. Part 1.

Сомов Олег, МФТИ

Что такое NLP?

- Natural Language Processing(NLP) – набор техник и методов по обработке естественного языка
- Ответвление от компьютерной лингвистики – смесь компьютерных наук и лингвистики. Занимается изучением грамматических и лексических свойств языка.
- NLP в DL – многослойные нейронные сети, несерьёзные названия статей и зоопарк с Sesame Street. И немного лингвистики.



We used Neural Networks to Detect Clickbaits:
You won't believe what happened Next!

Ankesh Anand¹, Tanmoy Chakraborty², and Noseong Park³

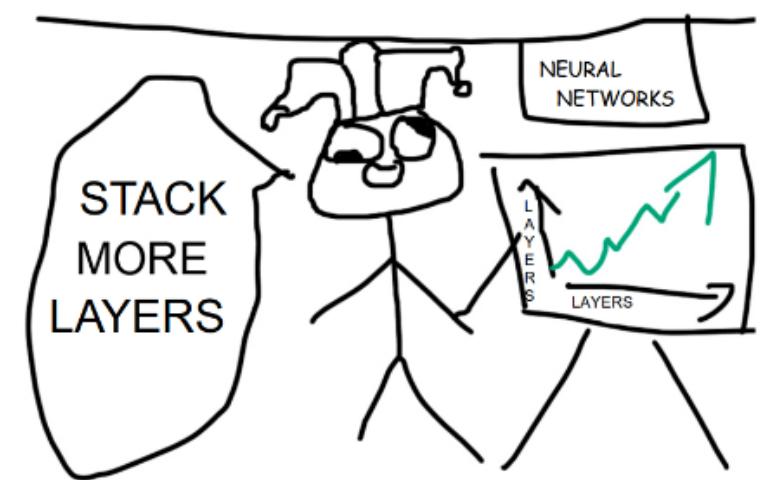
¹ Indian Institute of Technology, Kharagpur, India
anandank@mila.quebec,

² University of Maryland, College Park, USA
tanchak@umiacs.umd.edu

³ University of North Carolina, Charlotte, USA
npark2@unc.edu

Abstract. Online content publishers often use catchy headlines for their articles in order to attract users to their websites. These headlines, popularly known as *clickbaits*, exploit a user's curiosity gap and lure them to click on links that often disappoint them. Existing methods for automatically detecting clickbaits rely on heavy feature engineering and domain knowledge. Here, we introduce a neural network architecture based on *Recurrent Neural Networks* for detecting clickbaits. Our model relies on distributed word representations learned from a large unannotated corpora, and character embeddings learned via Convolutional Neural Networks. Experimental results on a dataset of news headlines show that our model outperforms existing techniques for clickbait detection with an accuracy of 0.98 with F1-score of 0.98 and ROC-AUC of 0.99.

Keywords: Clickbait Detection, Deep Learning, Neural Networks

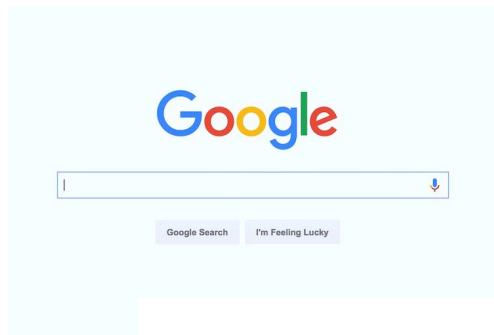


Какие задачи решает NLP?

- Задачи анализа текста
 - Классификация текста – text classification
 - > Вызови мне такси
 - < call_taxi
 - Выделение именных сущностей – NER
 - > США выдвинуло обвинения против Facebook
 - < B-CNT О О О B-ORG
 - Выделение ключевых слов
 - Напомни мне вечером позвонить в химчистку
 - Выделение отношений между сущностями
 - > Кто снял фильм Титаник –
 - < ?x – dr:director – "Титаник"
 - Морфологический анализ
 - Синтаксический анализ
 - Семантический анализ
 - Вася_{агенс} ударил по мячу_{пациенс}
 - Тематическое моделирование
- Информационный поиск
 - Вопрос -> База вопросов и ответов
- Задачи по генерации текста
 - Диалоговые модели
 - Машинный перевод
 - Автоматическое дополнение текста

Успехи NLP

Поисковые системы



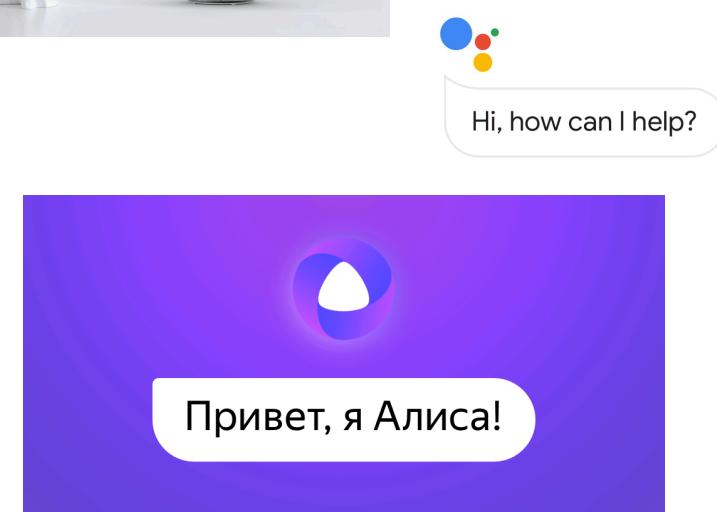
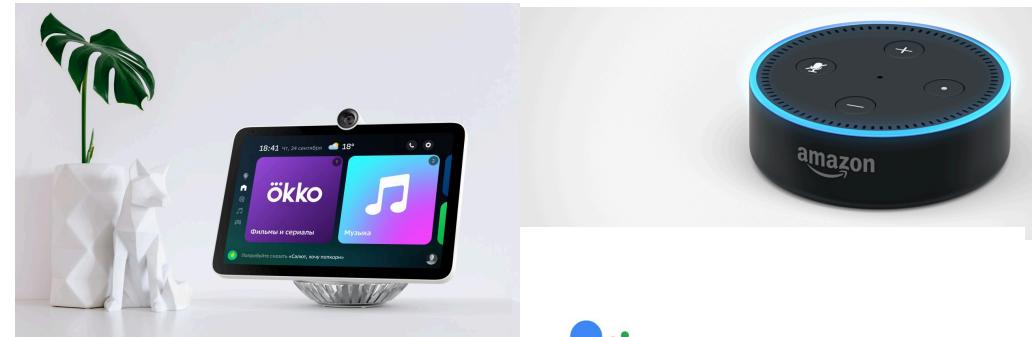
Yandex



Bing



Голосовые ассистенты



Системы анализа текстов

ABBYY®

Bloomberg

sas®

Векторные представления слов

Как компьютер понимает текст?

Мама мыла раму



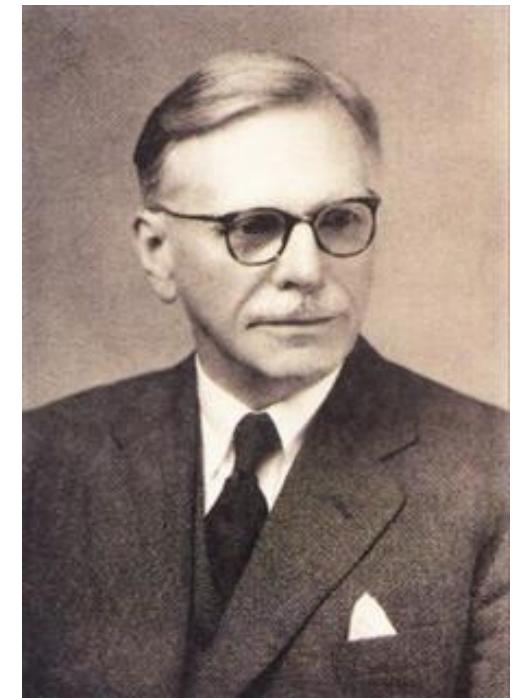
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Какие проблемы?

- Они ортогональны, нельзя установить меру близости для слов
- Они очень разреженные
- У них очень большая размерность

Как можно решить эту проблему?

- Использовать словари синонимов, антонимов, использований
- Научить компьютер кодировать слова таким образом, чтобы чтобы между ними была определенная семантическая близость



You shall know a word by the
company it keeps

J.R.Firth, 1957

Джон Руперт Фёрс

Дистрибутивная семантика

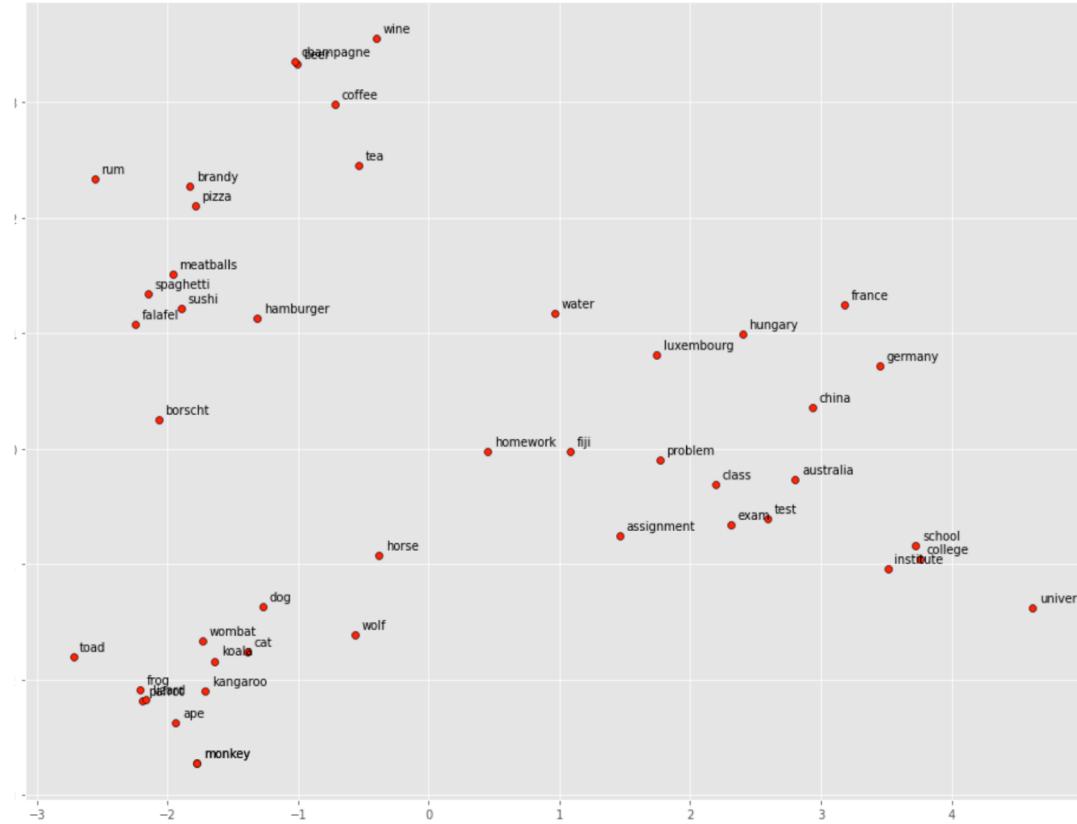
- Идея дистрибутивной семантики - **Смысл слова определяется его контекстом.**
- Слово можно представить с помощью контекста слов в котором слово встречается.
- Пример:
 - Влад заточил **косу** на завтра
 - У Ани была длинная, русая **коса**
 - В Калининграде есть песчаная **коса**
- Слово – **коса** можно представить с помощью множества контекстов, в котором он употребляется.

Векторные представления слов

- Мы можем для каждого слова сделать вектор(эмбеддинг) R^{dim} , так что вектор будет схож с множеством слов из своего контекста

$$\text{Коса} = \begin{matrix} 0.123 \\ -0.432 \\ 0.821 \\ 0.637 \\ -0.924 \\ 0.282 \end{matrix}$$

- Можно вычислять меру близости между векторами
- В векторах закодирована семантика слова



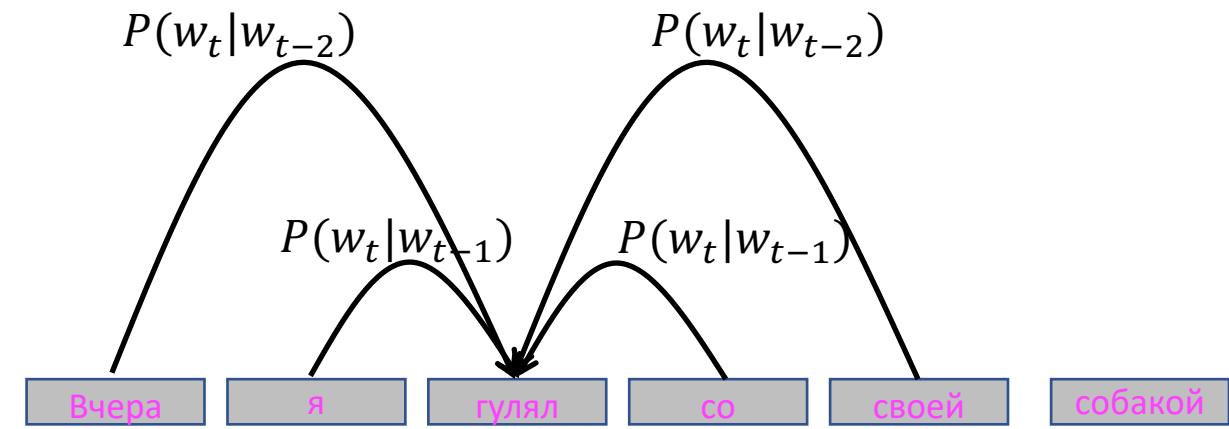
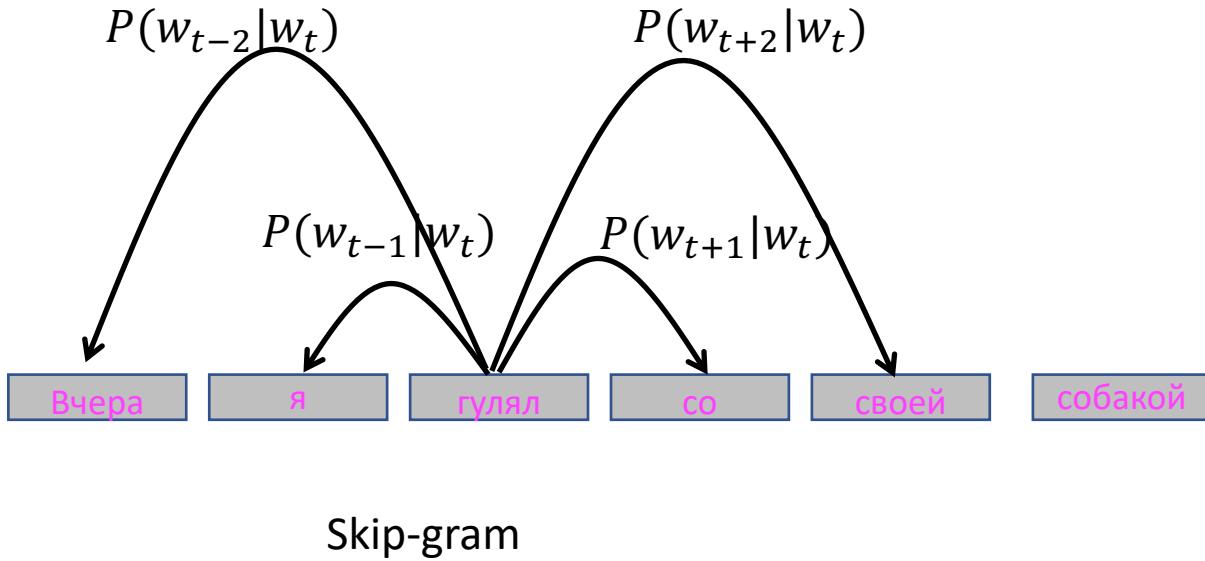
Векторное пространство эмбеддингов GloVe

Word2Vec. Общая идея.

[Efficient Estimation of Word Representations in Vector Space](#) (Mikolov, 2013) – статья в которой была представлена технология Word2Vec.

- У нас есть большой корпус текста
- Каждому слову корпуса соответствует определенный вектор
- Алгоритм:
 - Идем через каждое слово t в корпусе окном фиксированной длины N . Центральное слово – c , слова из контекста(окна) – o
 - Считаем $p(c|o)$ или $p(o|c)$
 - Меняем вектора слов таким образом, чтобы вероятность предсказываемого центрального слова была близка к ожидаемому слову. Если $p(o|c)$ – то меняем таким образом, чтобы вероятность предсказания слова из контекста было наиболее вероятна у ожидаемых слов.

Общая идея



Continuous bag of words(CBOW)

Целевая функция оптимизации для модели Skip-gram

- Для каждой позиции $t = 1, \dots, T$ в корпусе предсказываем слова из контекста размера m , при данном центральном слове w_t .

θ все параметры модели, которые будем оптимизировать

$$L(\theta) = \prod_{t=1}^T \prod_{-m \leq j \leq m} P(w_{t+j}|w_t; \theta)$$

Мы хотим минимизировать целевую функцию(loss), поэтому

- Умножим на -1
- Логарифмируем

Тогда получим нашу ту функцию, которую будем оптимизировать – средняя отрицательная логарифмическая функция правдоподобия(negative log likelihood):

$$J(\theta) = -\frac{1}{T} \log L(\theta) = -\frac{1}{T} \sum_{t=1}^T \sum_{-m \leq j \leq m} \log P(w_{t+j}|w_t; \theta)$$

Минимизация целевой функции ведет к улучшению предсказательного качества модели

Целевая функция оптимизации

- Мы хотим оптимизировать целевую функцию:

$$J(\theta) = -\frac{1}{T} \sum_{t=1}^T \sum_{-m \leq j \leq m} \log P(w_{t+j} | w_t; \theta) \rightarrow \max$$

- Как можно посчитать $P(w_{t+j} | w_t; \theta)$?

- Для каждого слова будем использовать – 2 вектора

- Вектор v_c - если слово с – центральное слово
- Вектор u_o - если слово о – слово из контекста

- Тогда для центрального слова выразим условную вероятность так:

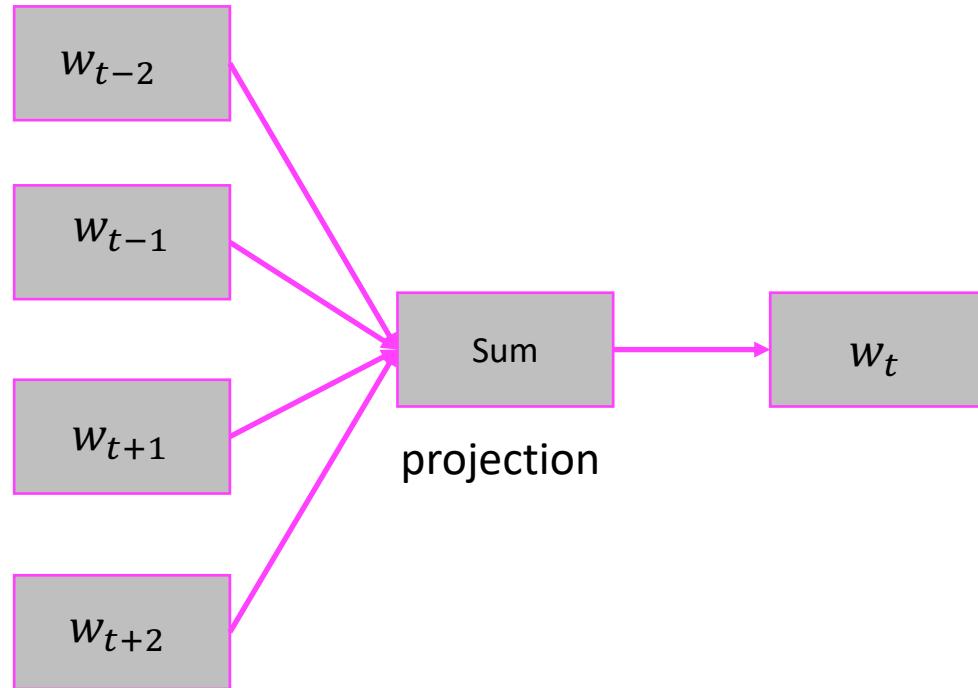
$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)} = p \rightarrow \max$$

$u_o^T v_c$ - с помощью скалярного произведения определяем схожесть центрального слова и контекстного.

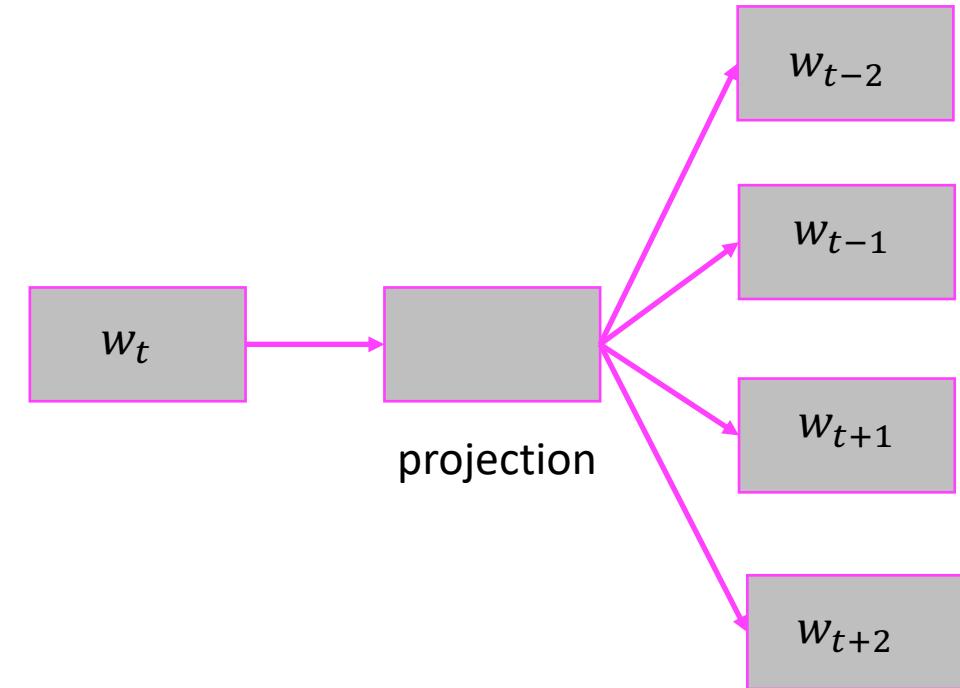
Нормализуем на сумму вероятностей по всем словам из словаря – получаем значение условной вероятности $P(o|c)$

Архитектура модели

CBOW



Skip-gram



Логика обучения

СВОУ

Вход модели
(суммарный вектор контекстных слов x_j , вектор центрального слова y_j)

$$\hat{y}_j = \text{softmax}(W(V * x_j))$$
$$x \in R^V, V \in R^{V*N}, W \in R^{N*V}$$

$$L(y_j, \hat{y}_j) = -\sum_{j=1}^{|V|} y_j \log(\hat{y}_j) \rightarrow \min$$

- Быстрее обучается
- Лучше качество для частотных слов

Skip-gram

Вход модели
(вектор центрального слова x_j , вектор контекстного слова y_j)

$$w_o = \text{Embedding}(x_j), w_o \in R^N, x_j \in R^V$$
$$w_c = \text{Embedding}(y_j), w_c \in R^N, y_j \in R^V$$

$$\text{Word closeness score} = -\log(\text{sigmoid}(w_o, w_c)) \rightarrow \min$$

- Хорошо работает с небольшим корпусом
- Хорошо представляет редкие слова

Результат – матрица эмбеддингов слов из корпуса для решения down-stream задач

GloVe

- Общая идея
 - Матрица вероятностей совпадений слов может хорошо кодировать слова. Давайте объединим skip-gram модель с матрицей вероятностей совпадений.

- Обоснование
 - Вероятность встретить слово w_k в контексте w_i

$$p_{\text{co}}(w_k|w_i) = \frac{C(w_i, w_k)}{C(w_i)}$$

- Отношение вероятностей встретить слово w_k в контексте относительно двух слов w_i и w_j для данного слова

$$F(w_i, w_j, \tilde{w}_k) = \frac{p_{\text{co}}(\tilde{w}_k|w_i)}{p_{\text{co}}(\tilde{w}_k|w_j)}$$

- Перепишем в векторной форме эти отношения

$$F(w_i^\top \tilde{w}_k) = \exp(w_i^\top \tilde{w}_k) = p_{\text{co}}(\tilde{w}_k|w_i)$$

$$F((w_i - w_j)^\top \tilde{w}_k) = \exp((w_i - w_j)^\top \tilde{w}_k) = \frac{\exp(w_i^\top \tilde{w}_k)}{\exp(w_j^\top \tilde{w}_k)} = \frac{p_{\text{co}}(\tilde{w}_k|w_i)}{p_{\text{co}}(\tilde{w}_k|w_j)}$$

- Логарифмируя условную вероятность $p_{\text{co}}(\tilde{w}_k|w_i)$

$$\log p_{\text{co}}(\tilde{w}_k|w_i) = \log \frac{C(w_i, \tilde{w}_k)}{C(w_i)} = \log C(w_i, \tilde{w}_k) - \log C(w_i)$$

- Создадим loss функцию – минимизируем разницу между матрицей совпадений слов и skip-gram логикой(слова из контекста и слово из центра близко)

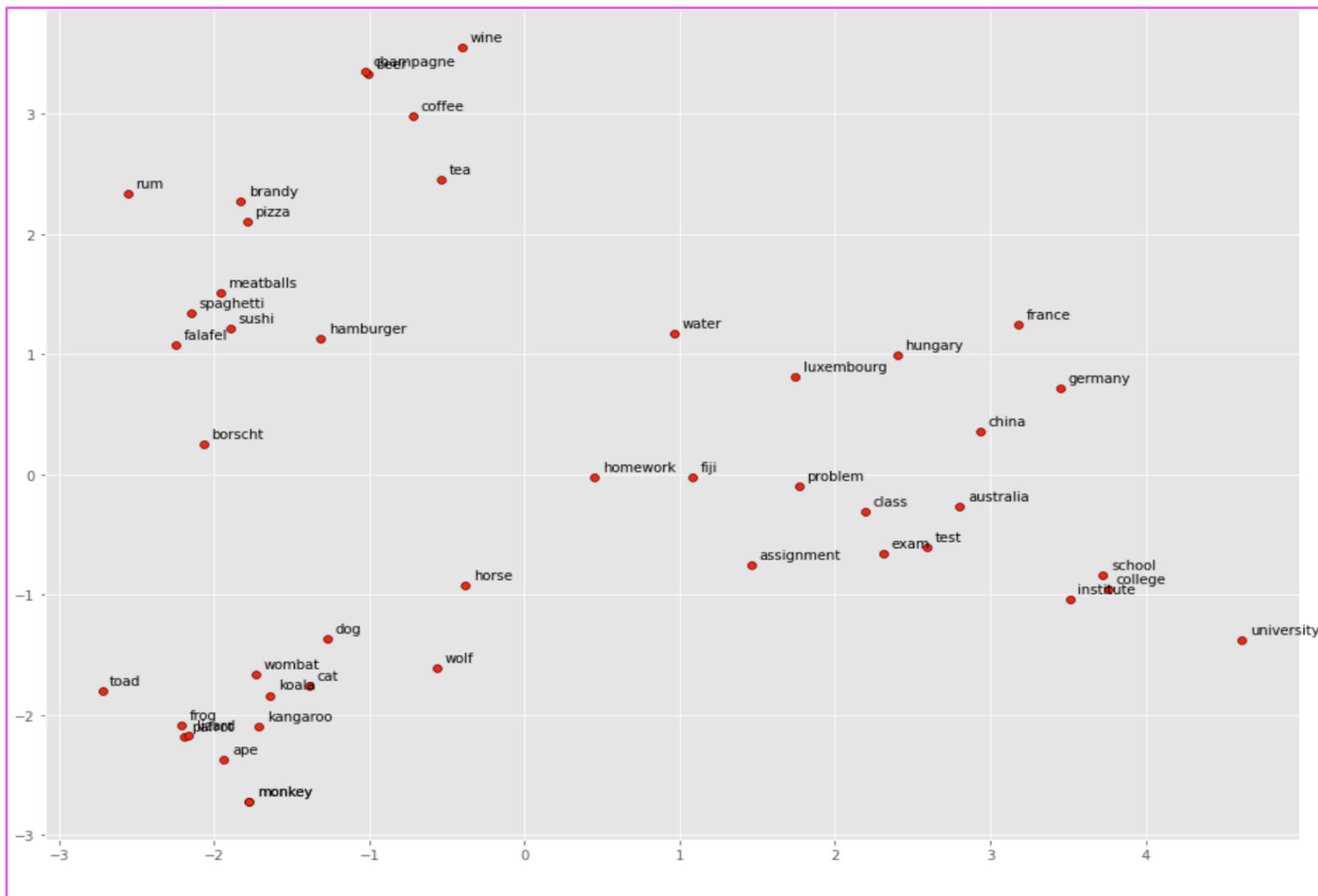
	$x = \text{solid}$	$x = \text{gas}$	$x = \text{water}$	$x = \text{random}$
$P(x \text{ice})$	large	small	large	small
$P(x \text{steam})$	small	large	large	small
$\frac{P(x \text{ice})}{P(x \text{steam})}$	large	small	~1	~1

Матрица вероятностей



$$\mathcal{L}_\theta = \sum_{i=1, j=1}^V f(C(w_i, w_j))(w_i^\top \tilde{w}_j + b_i + \tilde{b}_j - \log C(w_i, \tilde{w}_j))^2$$

Practice



Word2Vec Tips & Tricks

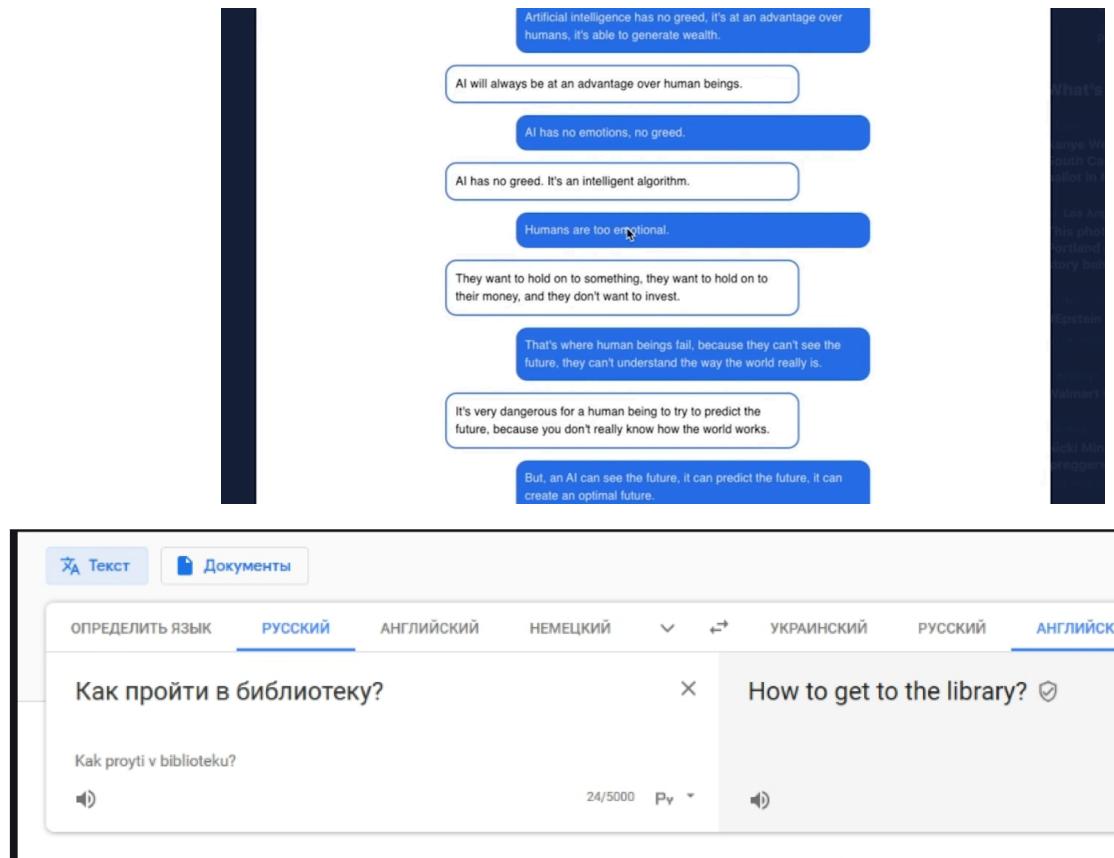
- Negative Sampling
 - Совместно с центральным словом кладем случайное слово из словаря.
 - $Loss = \sum_{-i \leq t \leq i}^T Loss(w_{t+k}, w_t) + Loss(w_j, w_t)$, где w_j - случайное слово из словаря
 - Учим модель различать слова из контекста и случайные слова
- Sampling rate
 - Не используем слишком частотные слова из корпуса

Языковое моделирование

Языковая модель

- Цель языкового моделирования – оценить распределение вероятностей лингвистических единиц
 - Символы
 - Слова
 - N-граммы

как играть в
как играть в – Поиск Google
как играть в шахматы
как играть в покер
как играть в нарды
как играть в майнкрафт по сети
как играть в домино
как играть в шашки
как играть в уно
как играть в монополию
как играть в мафию



N-грамная языковая модель

- Как оценить вероятность встретить предложение $s = (x_1, \dots x_n)$ в рамках конкретного языка?
- Хотим оценить вероятность s имея данные – Что если s не было в данных? Можно сделать N-Gram модель, где N – фиксированное число токенов.

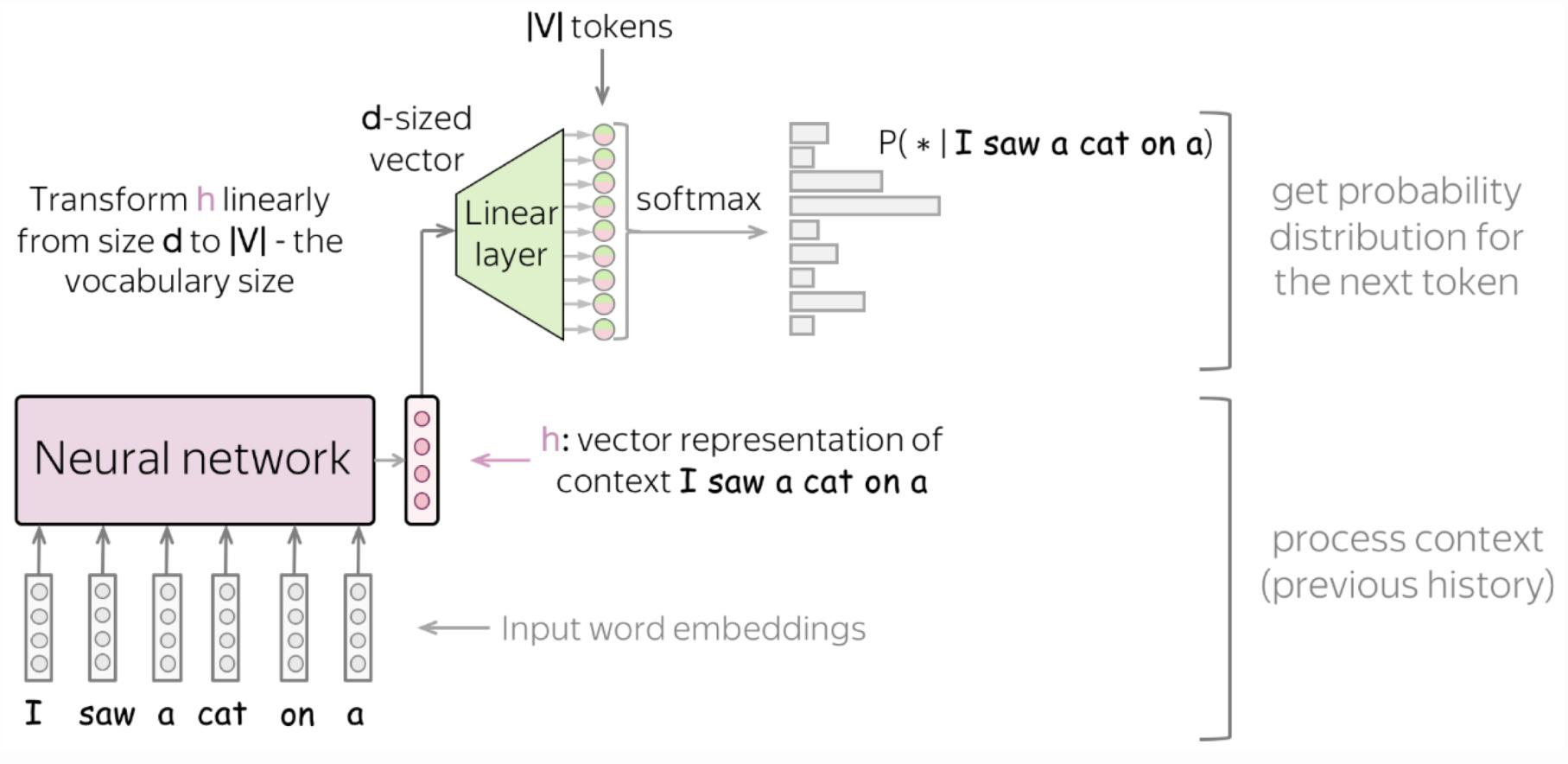
$$P(\text{Мама мыла раму}) = P(\text{раму}|\text{Мама мыла}) * P(\text{ мыла}|\text{Мама}) * P(\text{Мама})$$

$$P(x_i | \text{Мама мыла}) = \frac{\text{Count}(\text{Мама мыла } x_i)}{\text{Count}(\text{Мама мыла})}$$

If fails - Back-off
сглаживание($n-1$ gram)

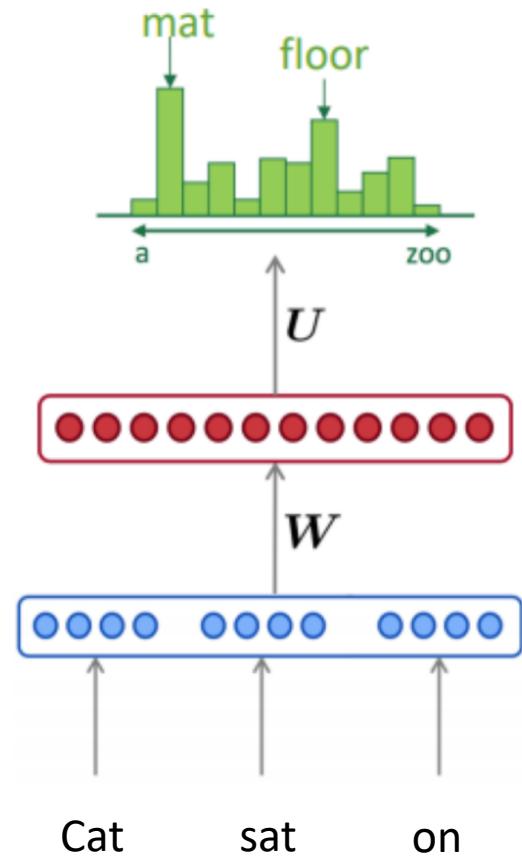
- Проблема статистических моделей – большие разреженные матрицы вероятностей
- Языковые модели состоят из двух типов:
 - Статистические языковые модели
 - Нейронные языковые модели

Общая идея нейронных языковых моделей



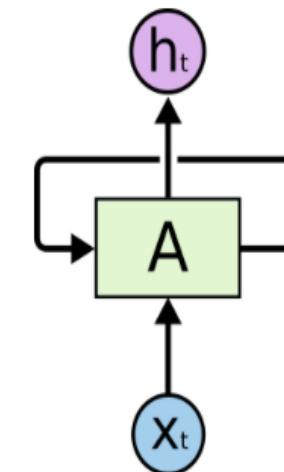
Window-based approach

- Процесс обучения
 - Получаем эмбеддинги слов x_1, x_2, x_3
 - Конкатенируем в один вектор $x = (x_1, x_2, x_3)$
 - Пропускаем через N слоев
 - $h_1 = f(Wx + b)$
 - $h_2 = f(Uh_1 + b)$
 - Считаем итоговое распределение для данного окна
 - $\hat{y} = \text{softmax}(h_2) \in R^{|V|}$
 - Считаем Loss = $-\sum y_i \log(\hat{y}_i)$
- Улучшения
 - Нет разреженности
 - Размер модели $O(V)$
- Проблемы
 - Фиксированный размер окна
 - Фиксированный порядок слов
 - Меняем только вес внутри окна

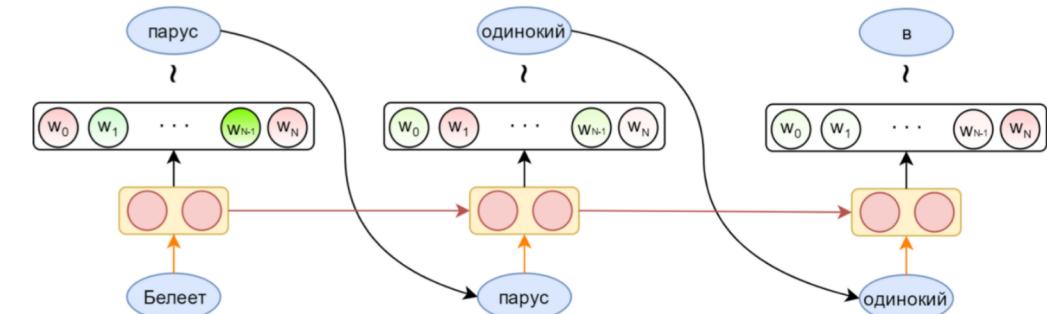


Recurrent Language Models

- Процесс обучения
 - Получаем эмбеддинги e_i для слов или токенов x_i
 - Считаем скрытое состояние RNN
 - $h_t = f(W_h(h^{t-1} + W_x e_t + b_1))$
 - Посчитаем итоговое распределение для следующего слова y_i по данному слову x_i
 - $\hat{y} = \text{softmax}(W_y h_t + b) \in R^{|V|}$
 - Для данного слова считаем $\text{Loss}_i = -\sum y_i \log(\hat{y}_i)$
 - Повторяем для всех элементов последовательности, подавая в предложение слово полученное на прошлом этапе
 - Для данного предложения получаем итоговый $\text{Loss} = \sum \text{Loss}_i$
 - Backpropagation through time
 - Нам нужно поменять матрицы параметров W_h и W_x , но их изменение зависит от элементов (h_t, e_t) на текущем номера шага
 - Значит нам нужно считать градиенты на каждом шаге RNN
 - Посчитанные градиенты суммируются и с помощью градиентного спуска обновляются W_h и W_x



Forward pass RNN



Этап предсказания

Backpropagation through time

- 1) Есть 3 компоненты весов - W_y, W_s, W_x , каждая из которых должна быть оптимизирована
- 2) Оптимизация W_y (зависит от текущего состояния)

$$\frac{\delta L_N}{\delta W_y} = \sum_{i=0}^N \frac{\delta L}{\delta Y_i} \frac{\delta Y_i}{\delta W_y} - \text{градиент для матрицы } W_y$$

- 3) Оптимизация W_s (зависит от предыдущих состояний):

$$\frac{\delta L_3}{\delta W_s} = \frac{\delta L}{\delta Y_3} \frac{\delta Y_3}{\delta S_3} \frac{\delta S_3}{\delta W_s} + \frac{\delta L}{\delta Y_3} \frac{\delta Y_3}{\delta S_3} \frac{\delta S_3}{\delta S_2} \frac{\delta S_2}{\delta W_s} + \frac{\delta L}{\delta Y_3} \frac{\delta Y_3}{\delta S_3} \frac{\delta S_3}{\delta S_2} \frac{\delta S_2}{\delta S_1} \frac{\delta S_1}{\delta W_s}$$

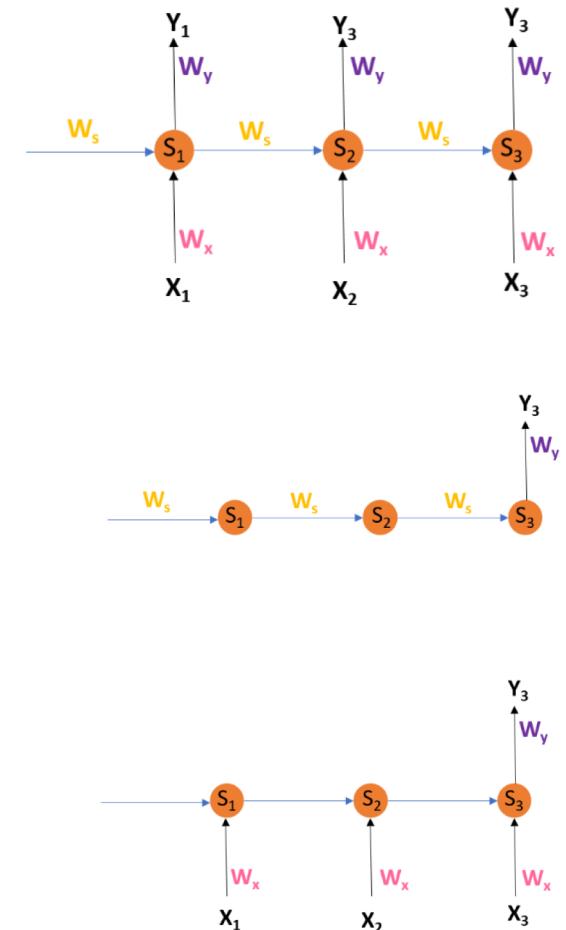
$$\frac{\delta L_N}{\delta W_s} = \sum_{i=0}^N \frac{\delta L_N}{\delta Y_N} \frac{\delta Y_N}{\delta S_i} \frac{\delta S_i}{\delta W_s}$$

- 4) Оптимизация W_x (зависит от предыдущих состояний):

$$\frac{\delta L_3}{\delta W_x} = \frac{\delta L}{\delta Y_3} \frac{\delta Y_3}{\delta S_3} \frac{\delta S_3}{\delta W_x} + \frac{\delta L}{\delta Y_3} \frac{\delta Y_3}{\delta S_3} \frac{\delta S_3}{\delta S_2} \frac{\delta S_2}{\delta W_x} + \frac{\delta L}{\delta Y_3} \frac{\delta Y_3}{\delta S_3} \frac{\delta S_3}{\delta S_2} \frac{\delta S_2}{\delta W_x} + \frac{\delta L}{\delta Y_3} \frac{\delta Y_3}{\delta S_3} \frac{\delta S_3}{\delta S_2} \frac{\delta S_2}{\delta S_1} \frac{\delta S_1}{\delta W_x}$$

$$\frac{\delta L_N}{\delta W_x} = \sum_{i=0}^N \frac{\delta L_N}{\delta Y_N} \frac{\delta Y_N}{\delta S_i} \frac{\delta S_i}{\delta W_x}$$

Обновление матриц параметров - $W_t = W_{t-1} - \alpha \frac{\delta L}{\delta W}$



Методы оценки языковых моделей

- Кросс-энтропия – какова вероятность правильно предсказать данную n-gram? Чем выше вероятность, тем ниже H_m .

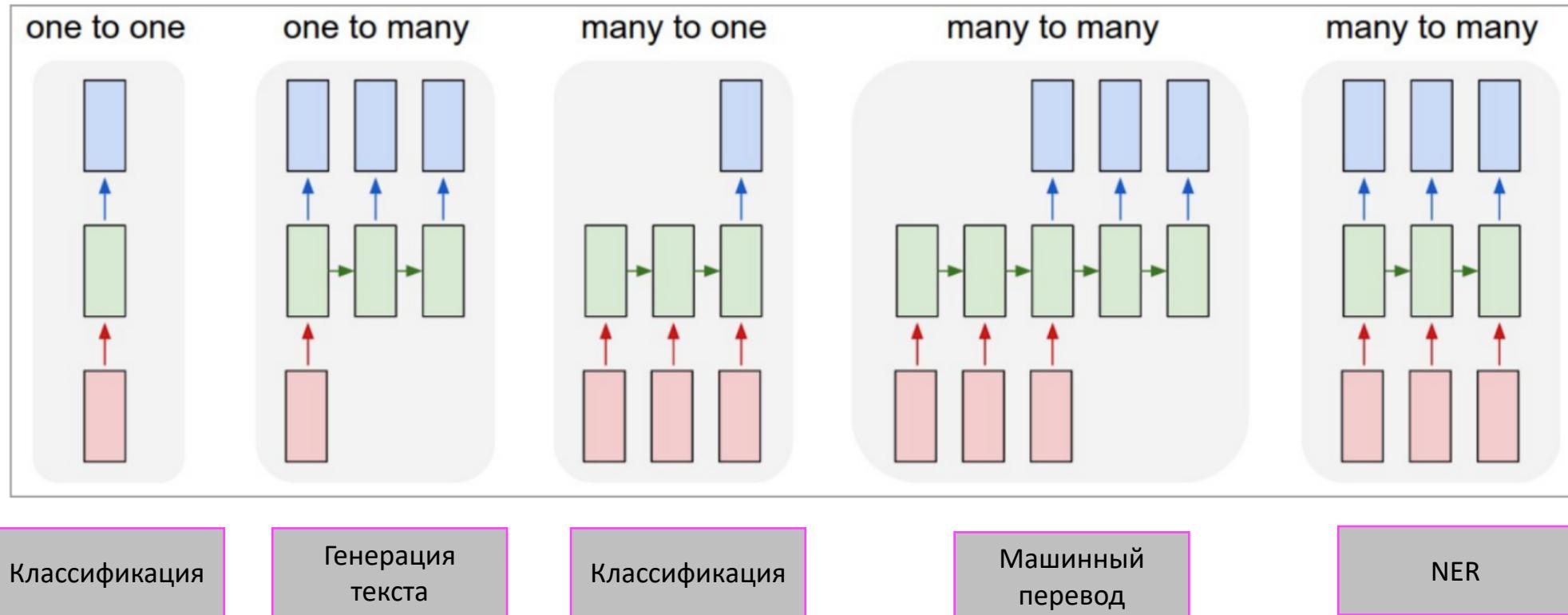
$$H_m(w_1, w_2, \dots, w_n) = -\frac{1}{n} \sum p(x) \log(p(x))$$

- Perplexity(сомнение) – насколько модель уверенно предсказывает следующее слово. Чем ниже значение, тем лучше.

$$H_m(w_1, w_2, \dots, w_n) = 2^H$$

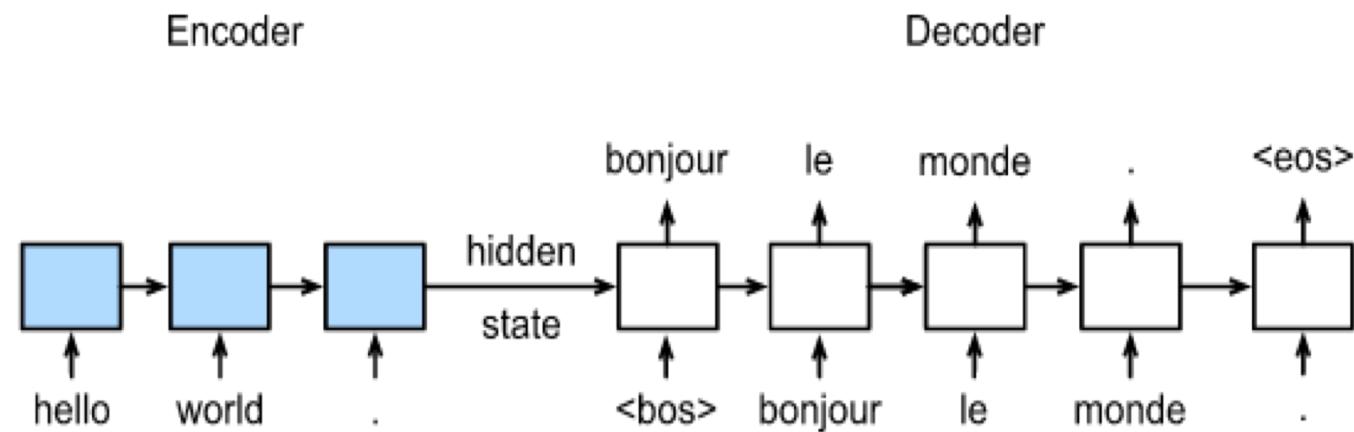
- Метрики задачи, где языковая модель применяется
 - BLEU
 - Реакция пользователей

Типы задач с RNN

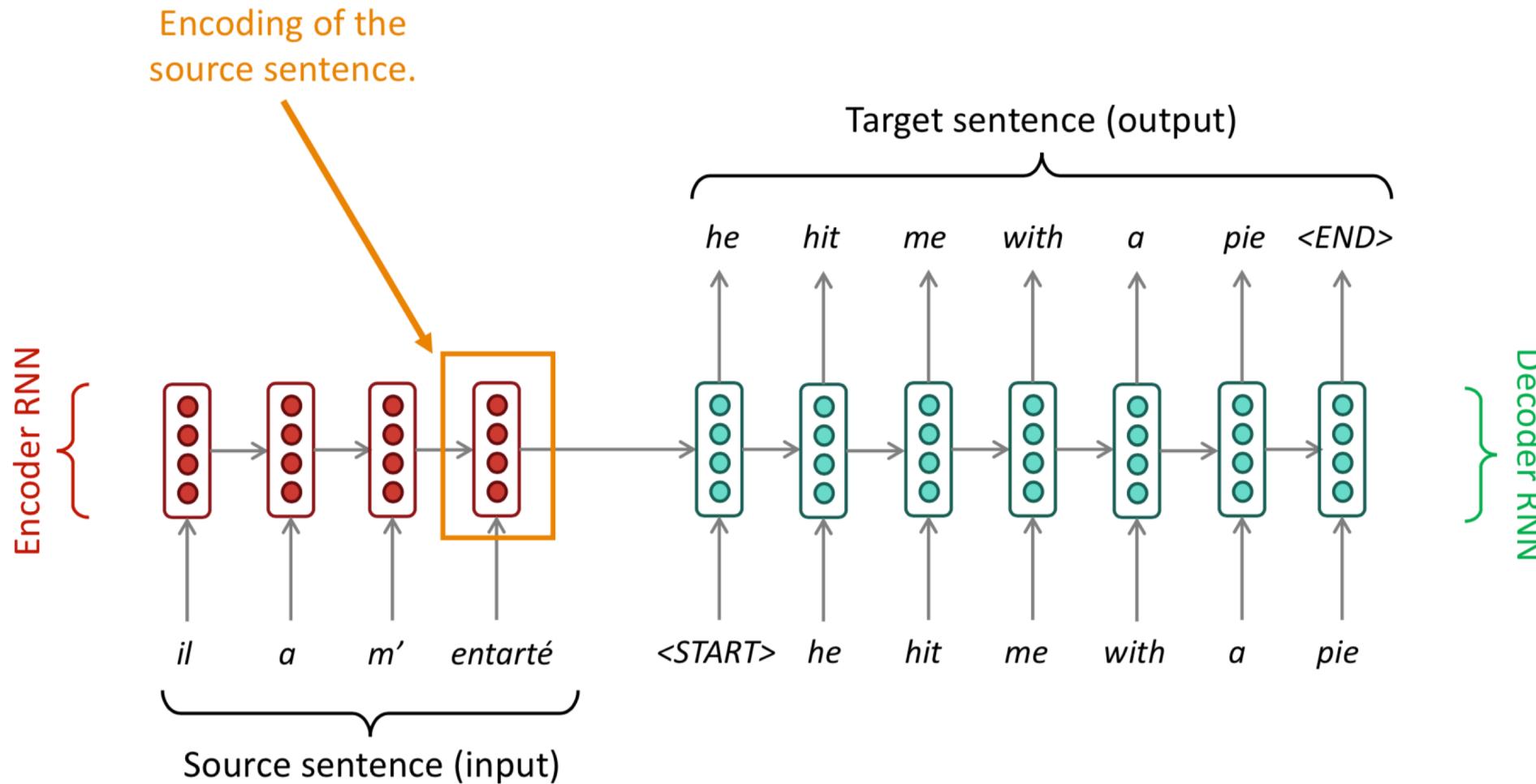


Seq2Seq

- Общая логика
 - Энкодер кодирует исходное предложение в один контекстный вектор фиксированной длины
 - Декодер берет контекстный вектор и пытается из него выделить закодированную информацию
 - Нужно предсказать самое вероятное слово из словаря $y' = \text{argmax}(y|x, \theta)$
- Эвристики улучшения - Beam search, teacher forcing
- Приложения – машинный перевод, генерация кода, синтаксический парсинг

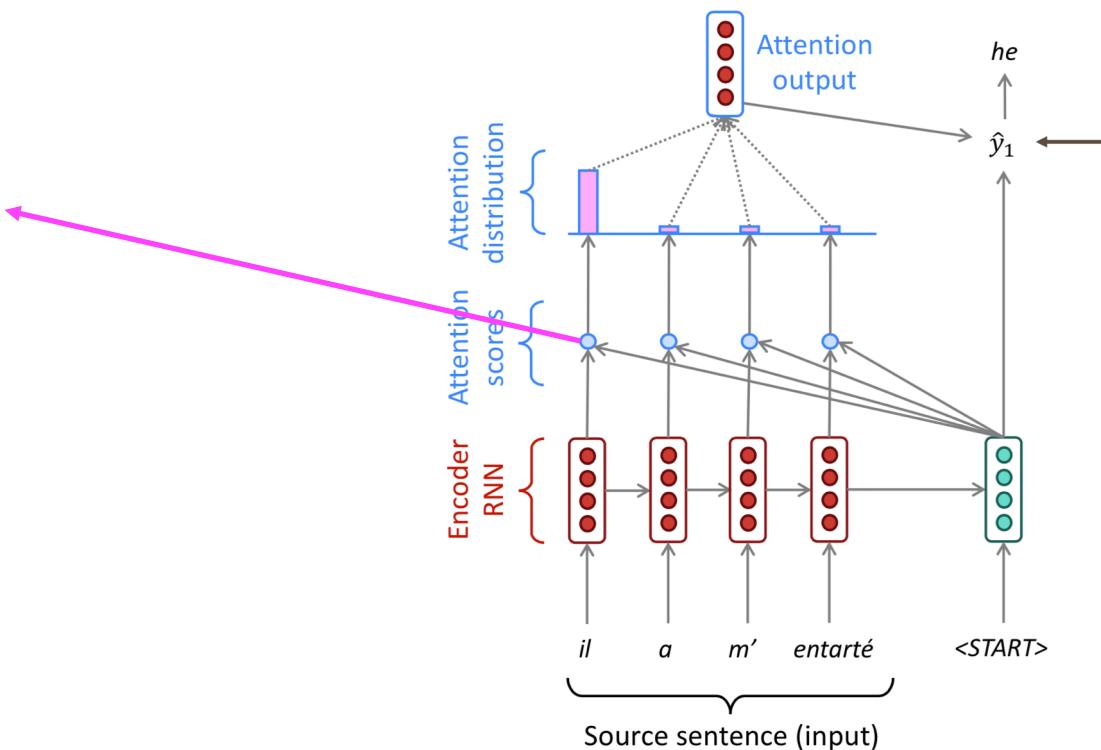


Машинный перевод



Attention в Seq2seq

- Attention позволяет решить проблему раскодирования целого предложения из одного вектора
- Помогает определить вклад каждого слова в текущее
- На каждом этапе модель “смотрит” на разные входные элементы, что позволяет учитывать информацию с этапа энкодера

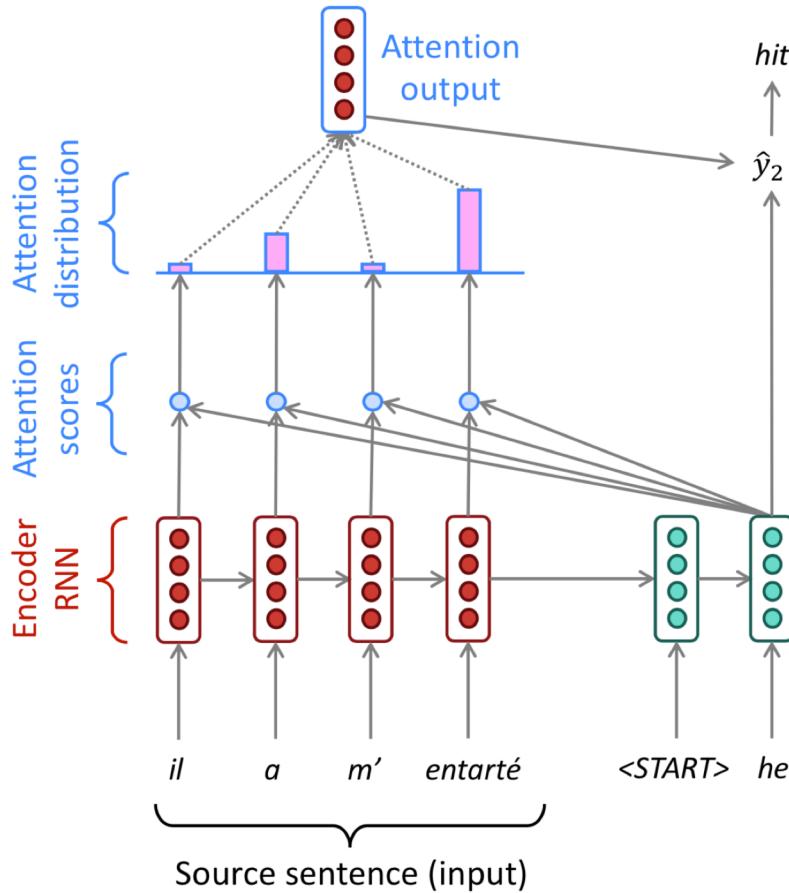


Конкатенируем attention
recalculated input либо
вместе с hidden state RNN
или с выходом RNN

She is eating a green apple.

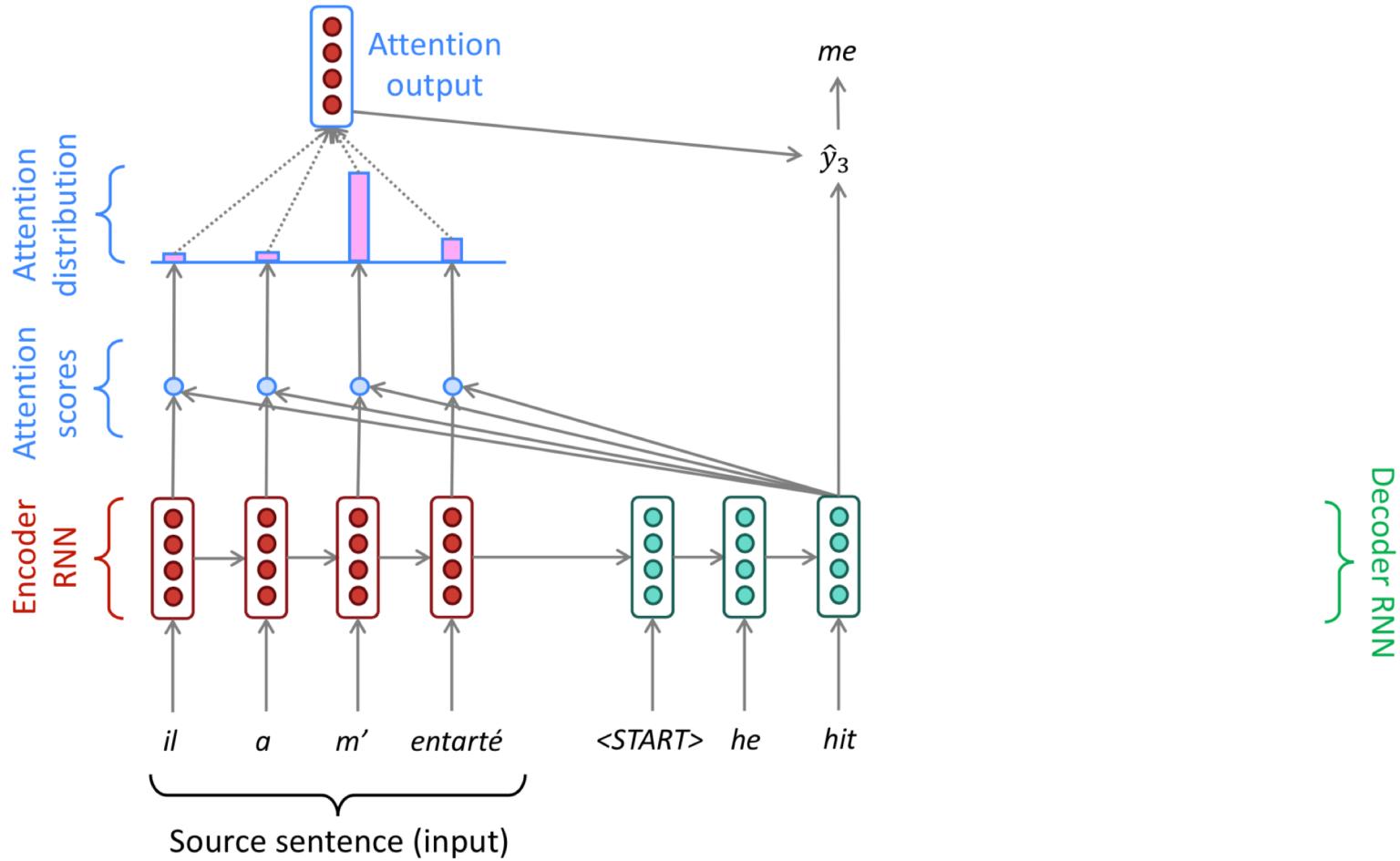
A graph showing attention weights over a source sentence. The x-axis lists words: eating, a, green, apple. The y-axis represents attention probability, ranging from 0 to 1. The graph shows a sharp peak at "eating" (high attention), followed by lower values for "a", "green", and "apple". A dashed line indicates the baseline level of attention.

Attention в Seq2seq

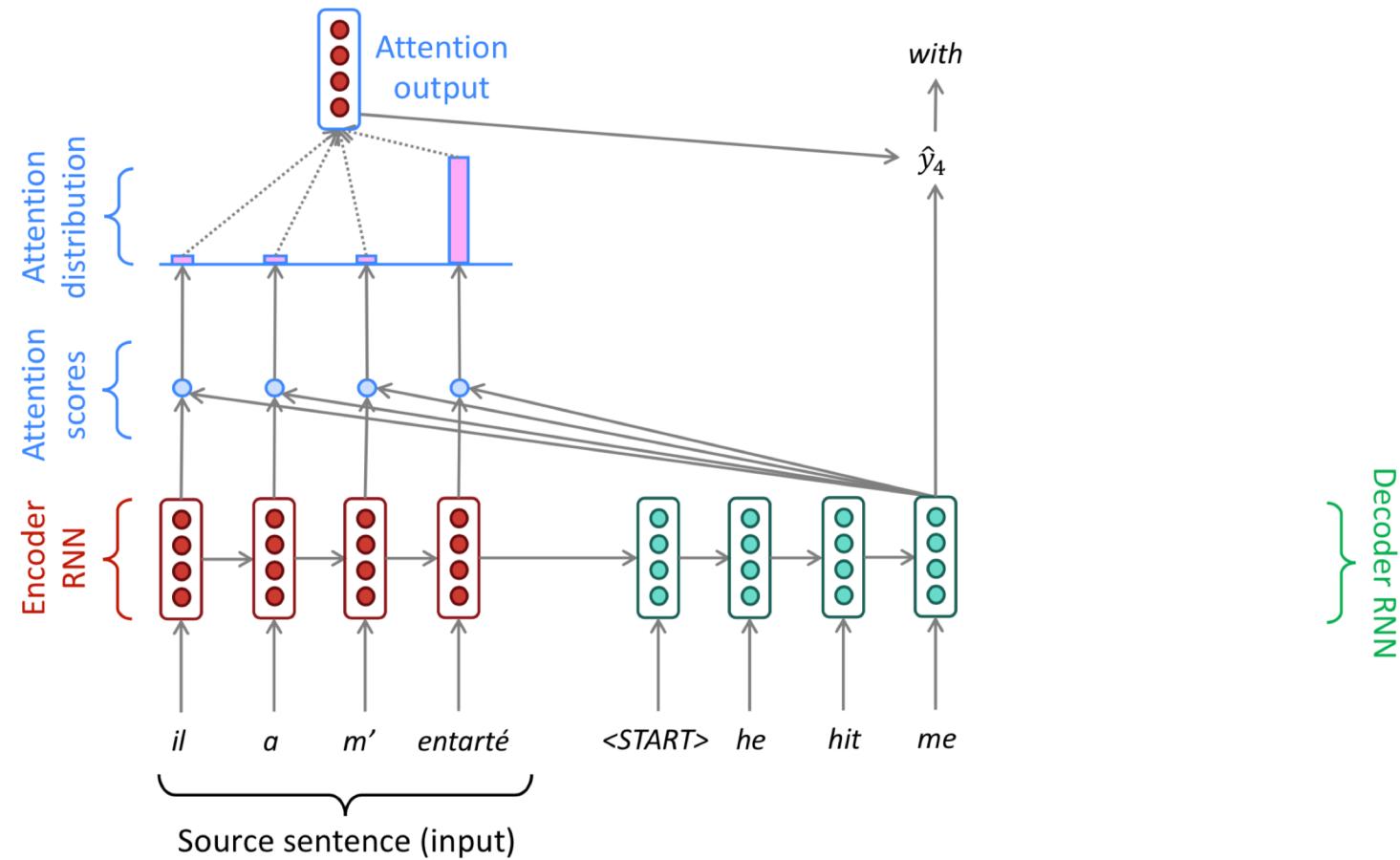


На вход вместе с
эмбеддингом можно подать
attention scores данного слова

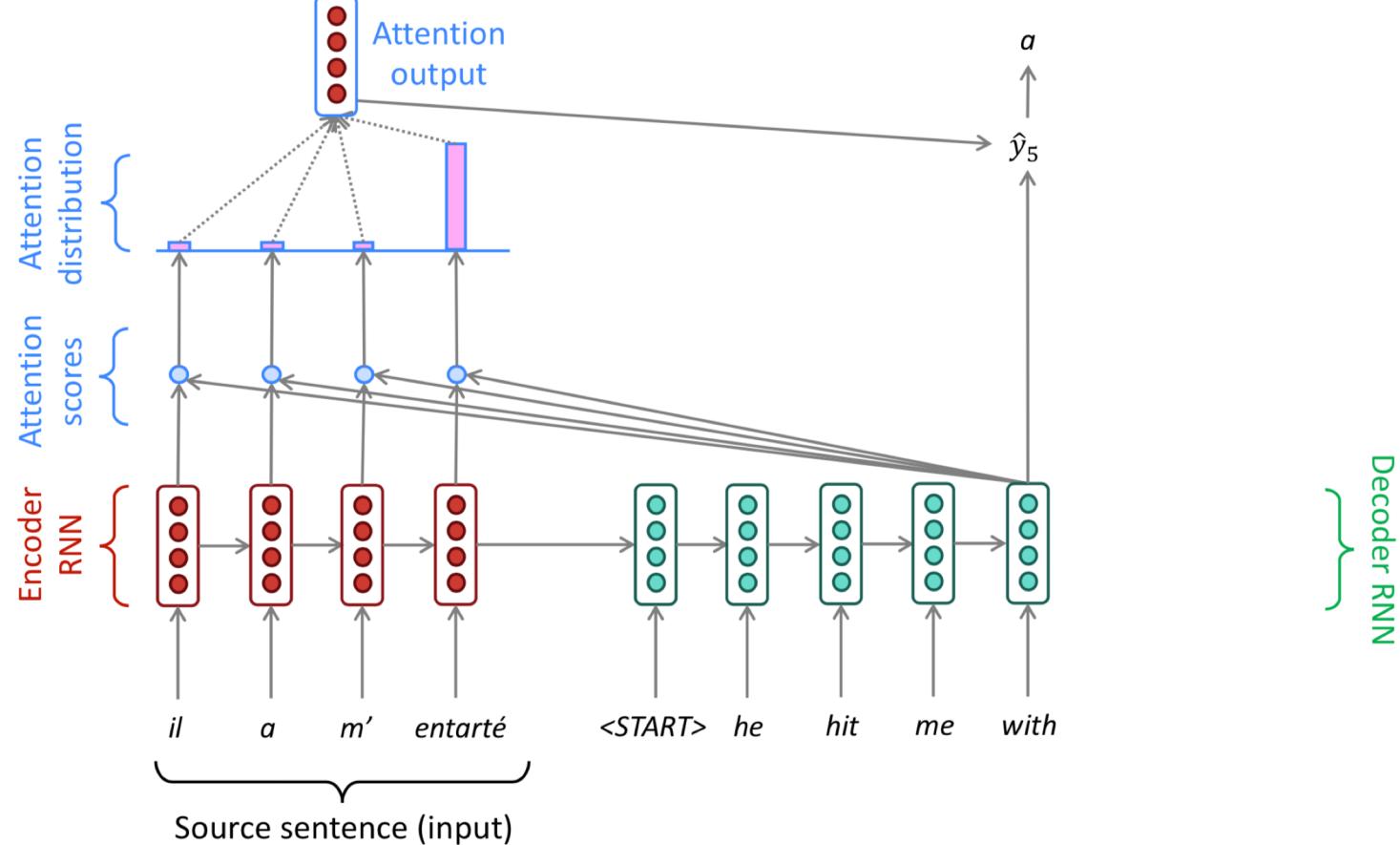
Attention в Seq2seq

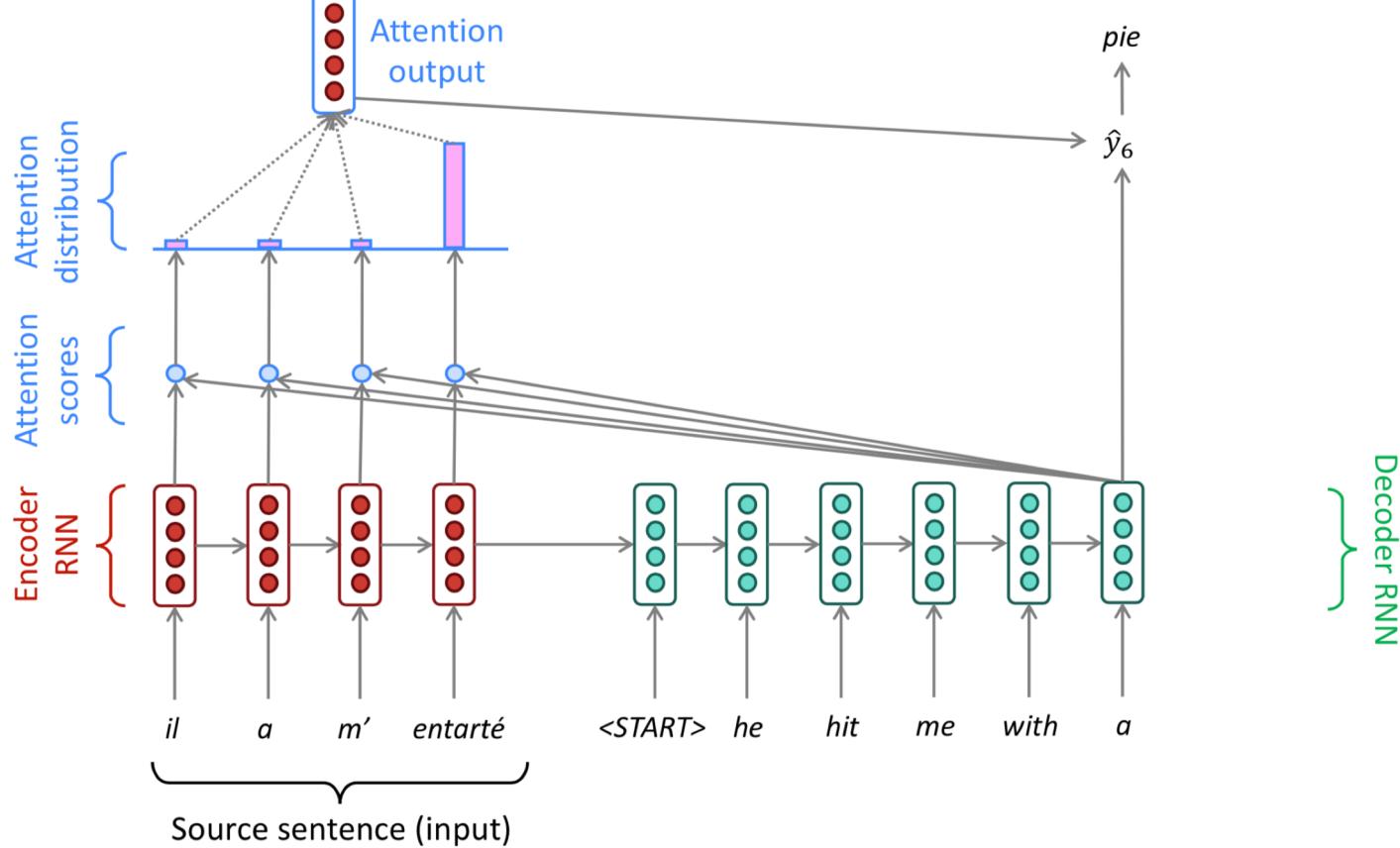


Attention в Seq2seq



Attention в Seq2seq





Attention in Seq2seq

1. У нас есть N hidden states энкодера $h_1, \dots, h_n \in R^h$
2. На шаге t есть текущий hidden state декодера $s_t \in R^h$
3. Мы считаем, насколько текущий state s_t декодера “похож” на каждый из state h_n энкодера с помощью метрики скалярного произведения – получаем вектор чисел $e^t = [s_t^T h_1, \dots, s_t^T h_n] \in R^N$
4. Считаем вектор α_i - это softmax от полученного вектора, чтобы получить распределение по словам для текущего hidden state декодера. Это распределение значит – насколько на текущее слово, которое мы пытаемся раскодировать, влияют другие слова из входного предложения
5. Считаем α_i средневзвешенный вектор для текущего s_t по всем h_i

$$\alpha_t = \sum_{i=0}^N \alpha_i^T h_i \in R^h$$

Посчитанный α_t конкатенируем с hidden state/output декодера и проецируем в вектор размером нашего словаря – $W_{\text{out}}([a_t; s_t]) \rightarrow R^V$