Seq2Seq / Encoder-Decoder / Attention

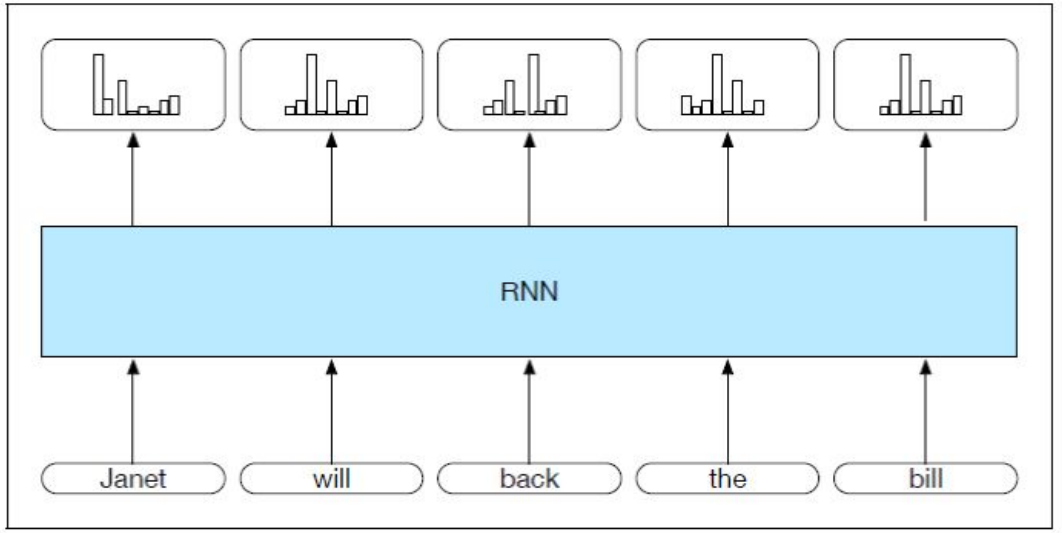
Преподаватель: Герард Костин

План

- **Encoder Decoder**
- Attention
- Self-Attention

Encoder-Decoder

- **RNN:**входная последовательность преобразуется в выходную последовательность взаимно однозначным образом (one2one).



- **Цель:** разработать архитектуру, способную генерировать соответствующие контексту выходные последовательности произвольной длины.

  Приложения:

  - **Машинный перевод,**
  - **Обобщение,**
  - **Вопрос-Ответ,**
  - **Диалоги.**

-

Простая RNN сеть

Наиболее существенное изменение: новый набор весов U подключает скрытый слой из предыдущего временного шага к текущему скрытому слою.

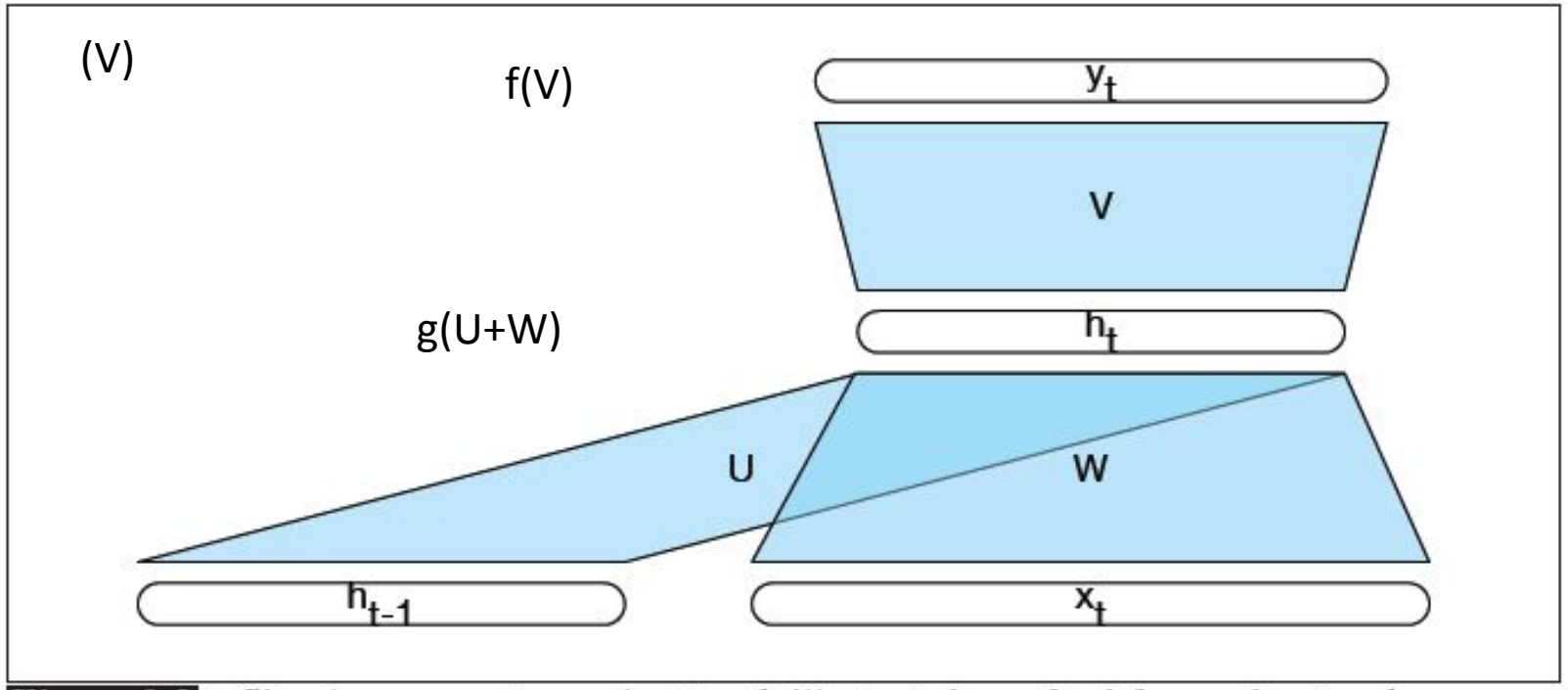
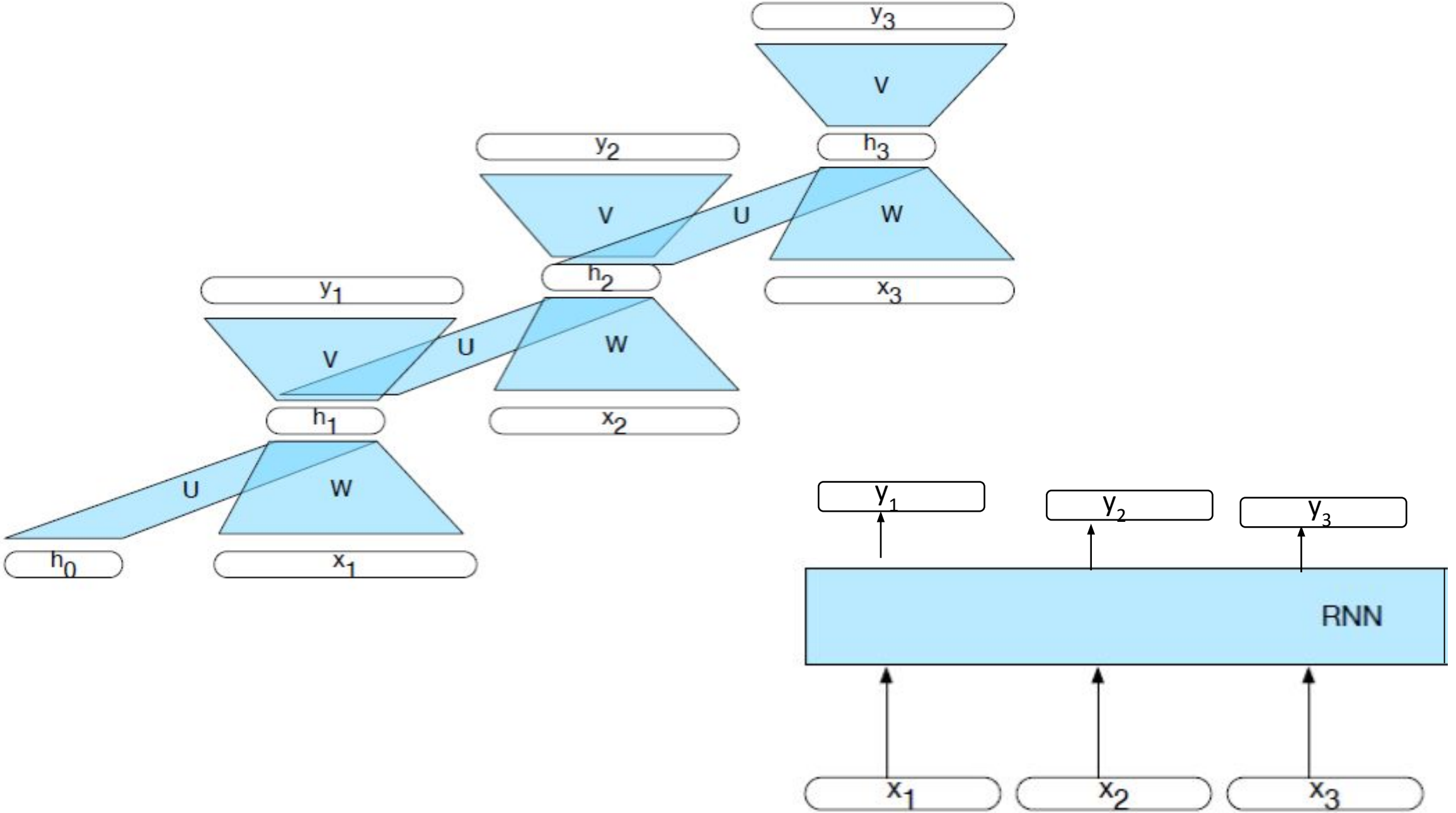
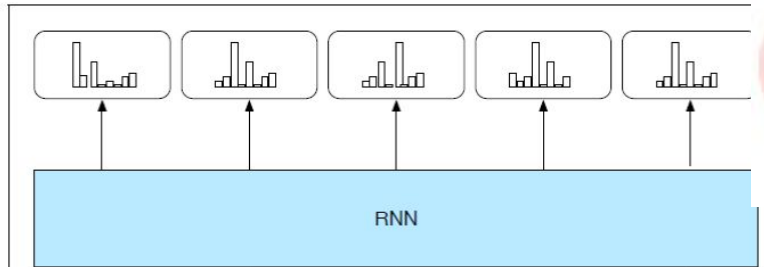
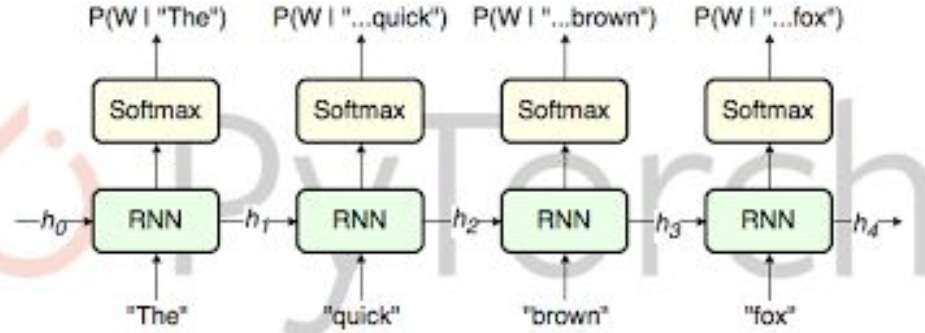


Figure 9.3 Simple recurrent neural network illustrated as a feed-forward network.

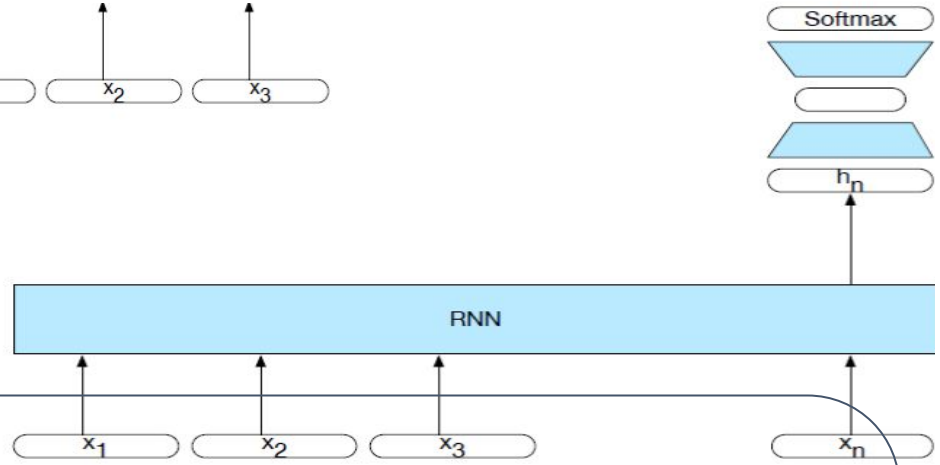

y3
V
h3
y2
V
U
W
h2
x3
y1
V
U
W
h1
x2
U
W
h0
x1
y1
y2
y3
RNN
x1
x2
x3

RNN Применение

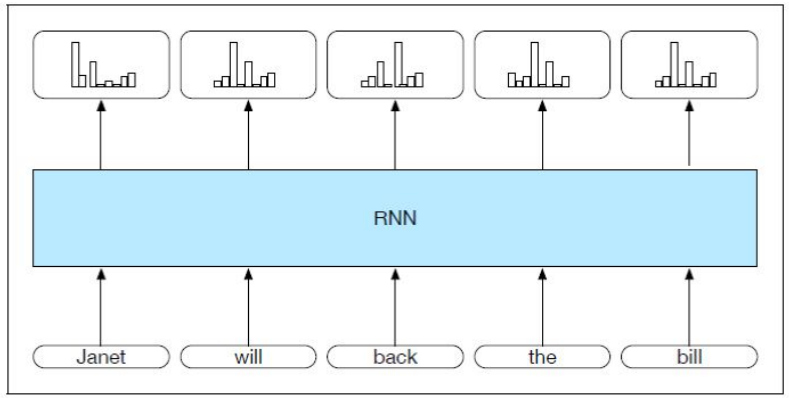




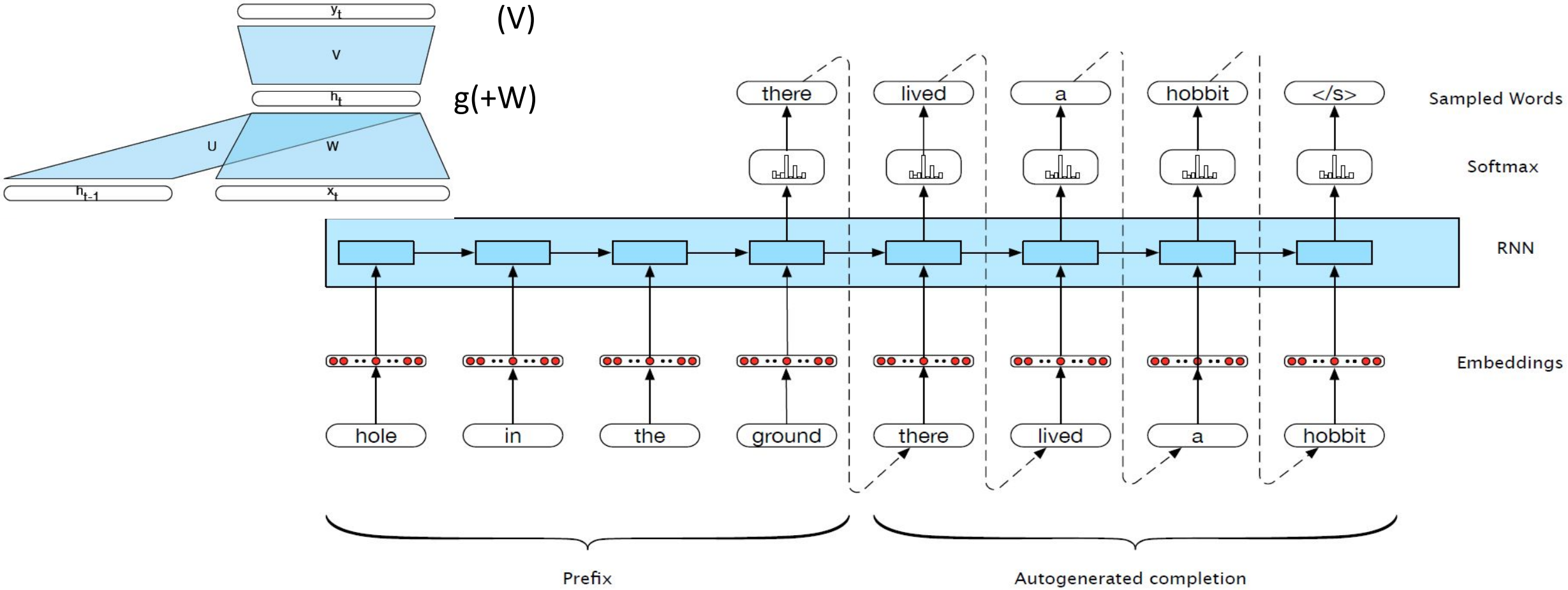
- Sequence Classification (Sentiment, Topic)



- Sequence to Sequence



Sentence Completion - RNN



- **Обученную модель можно использовать для создания новых последовательностей.**
- **Или для завершения заданной последовательности (пока не будет сгенерирован токен конца предложения <\ s>)**

Расширенная (авторегрессия) генерация - машинный перевод

слово, генерируемое на каждом временном шаге, зависит от слова из предыдущего шага.

- Данные обучения представляют собой параллельный текст, например, English / French

  there lived a hobbit *vivait un hobbit*

  ……..

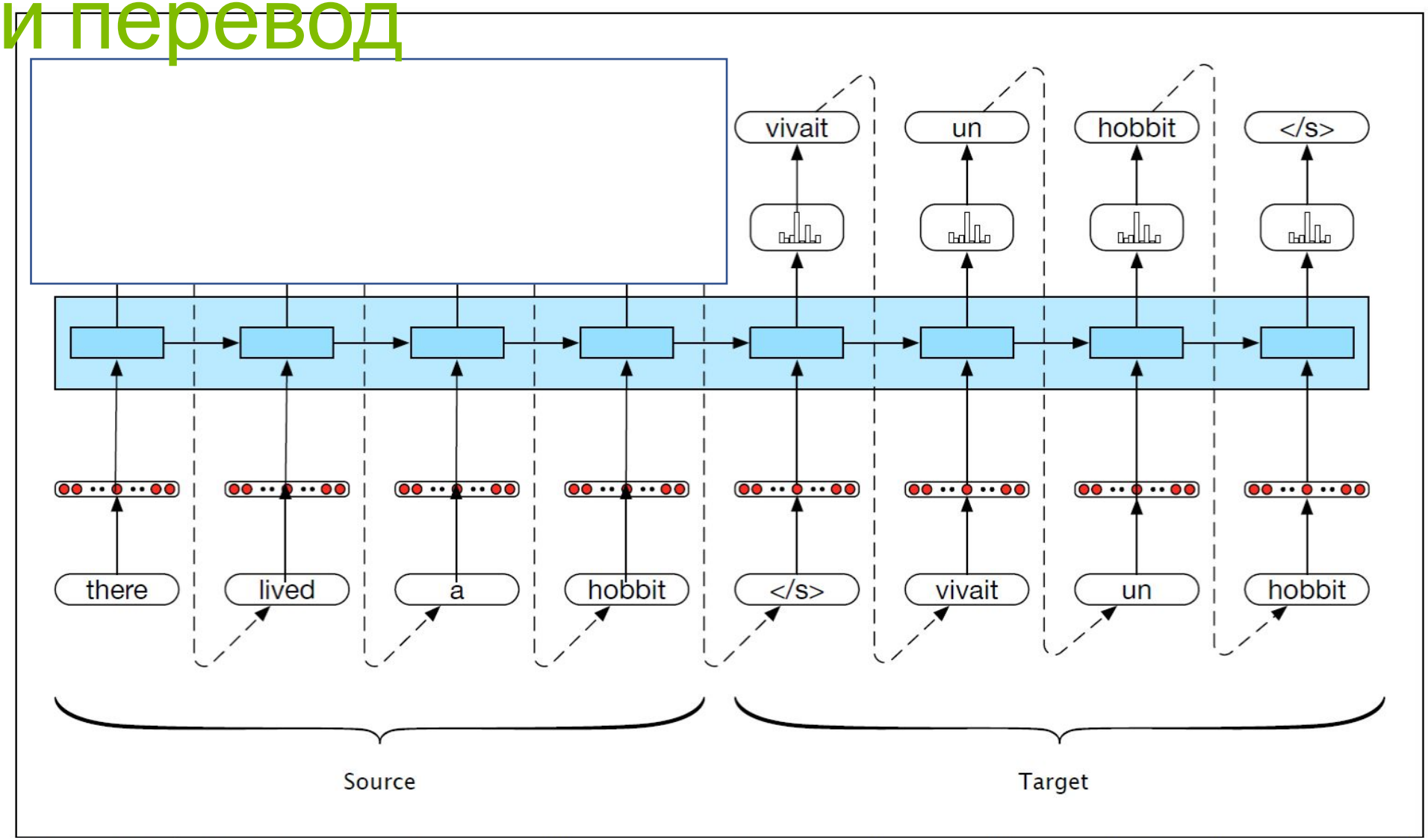
- Построить языковую модель RNN на объединении входа и выхода

  there lived a hobbit <\s> *vivait un hobbit* <\s>

  ……..

Расширенная (авторегрессия) генерация - машинный перевод

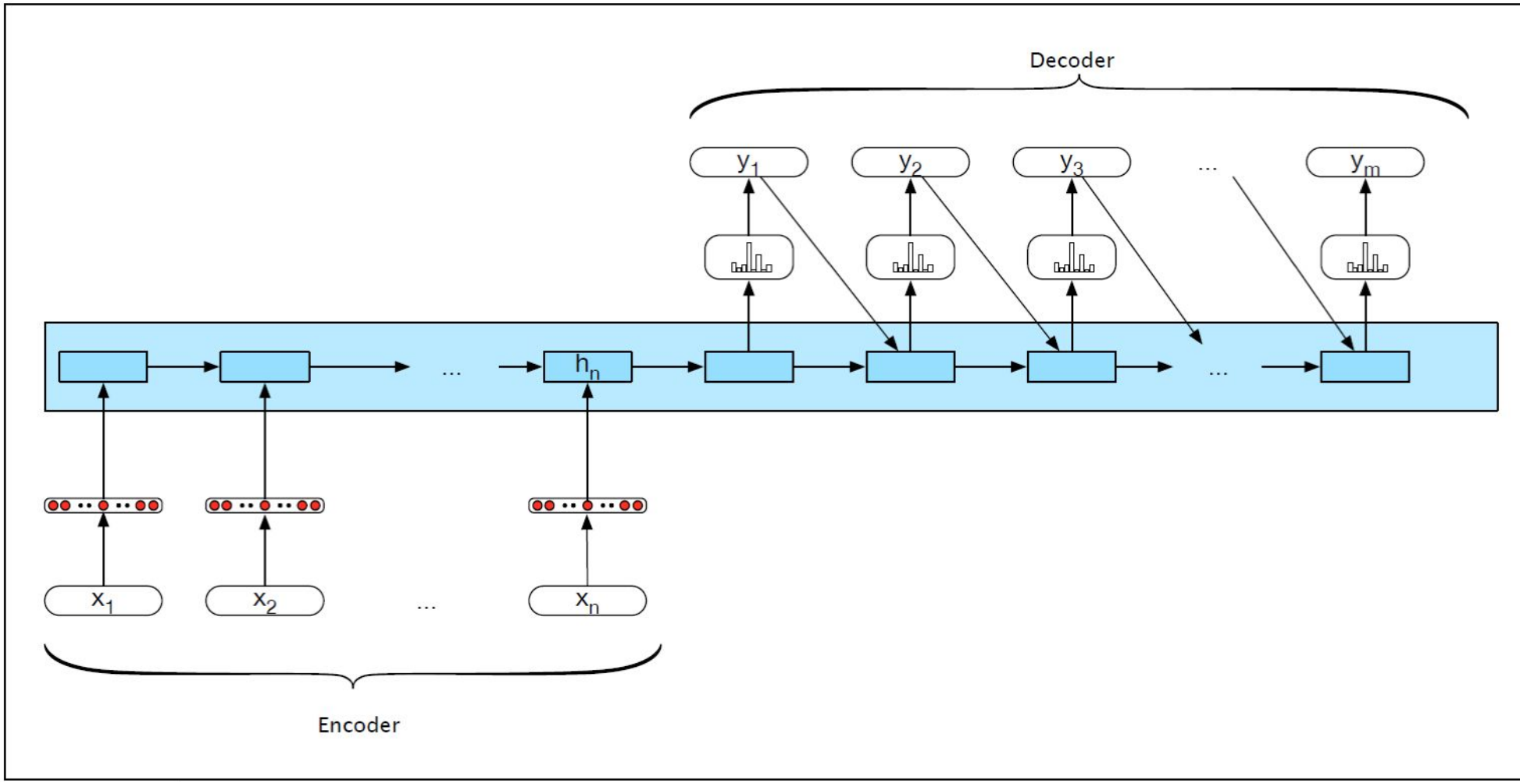
- Перевод как завершение предложения!



Encoder Decoder Networks

Ограничения

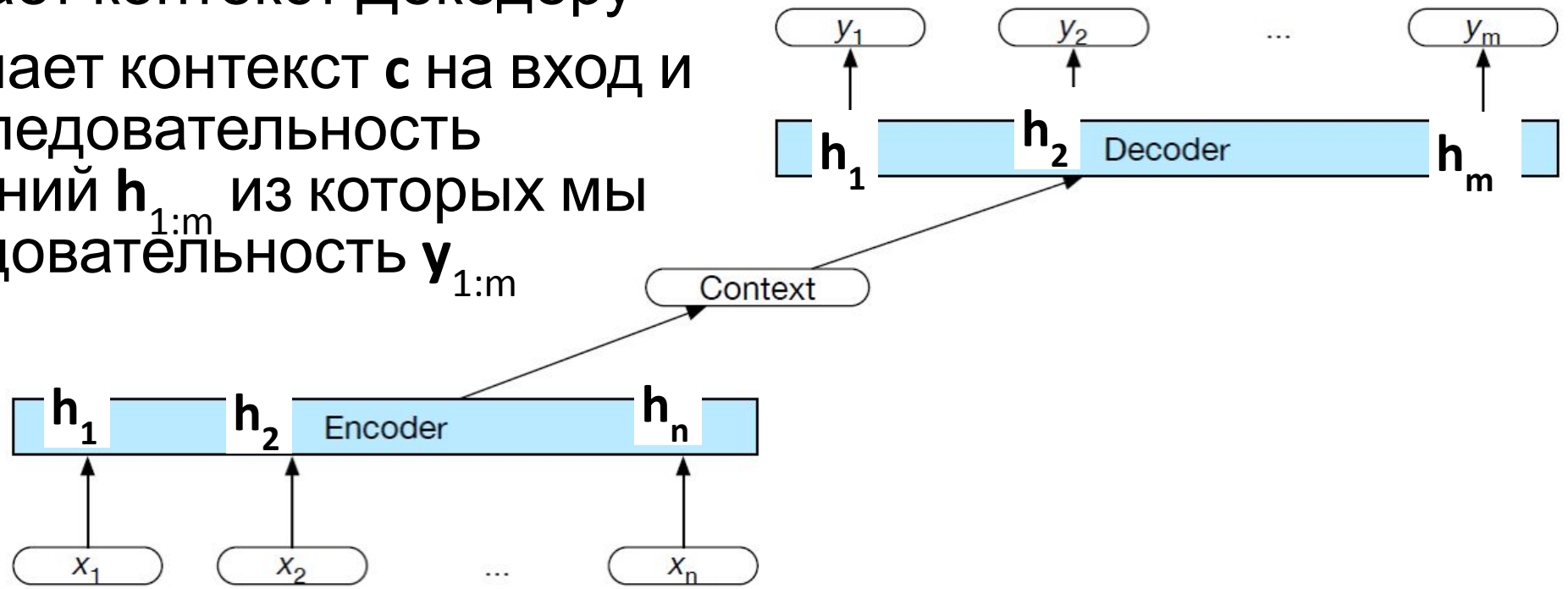
- **E** and **D** предполагается, что они имеют одинаковую внутреннюю структуру (RNN)
- Выход **E** единственная информация, доступная **D**
- Эта информация доступна **D** только на первом шаге.



- Encoder генерирует контекстное представление ввода (последнее состояние).
- Decoder принимает это состояние и авторегрессивно генерирует последовательность выходных данных

Основы Encoder Decoder Сетей

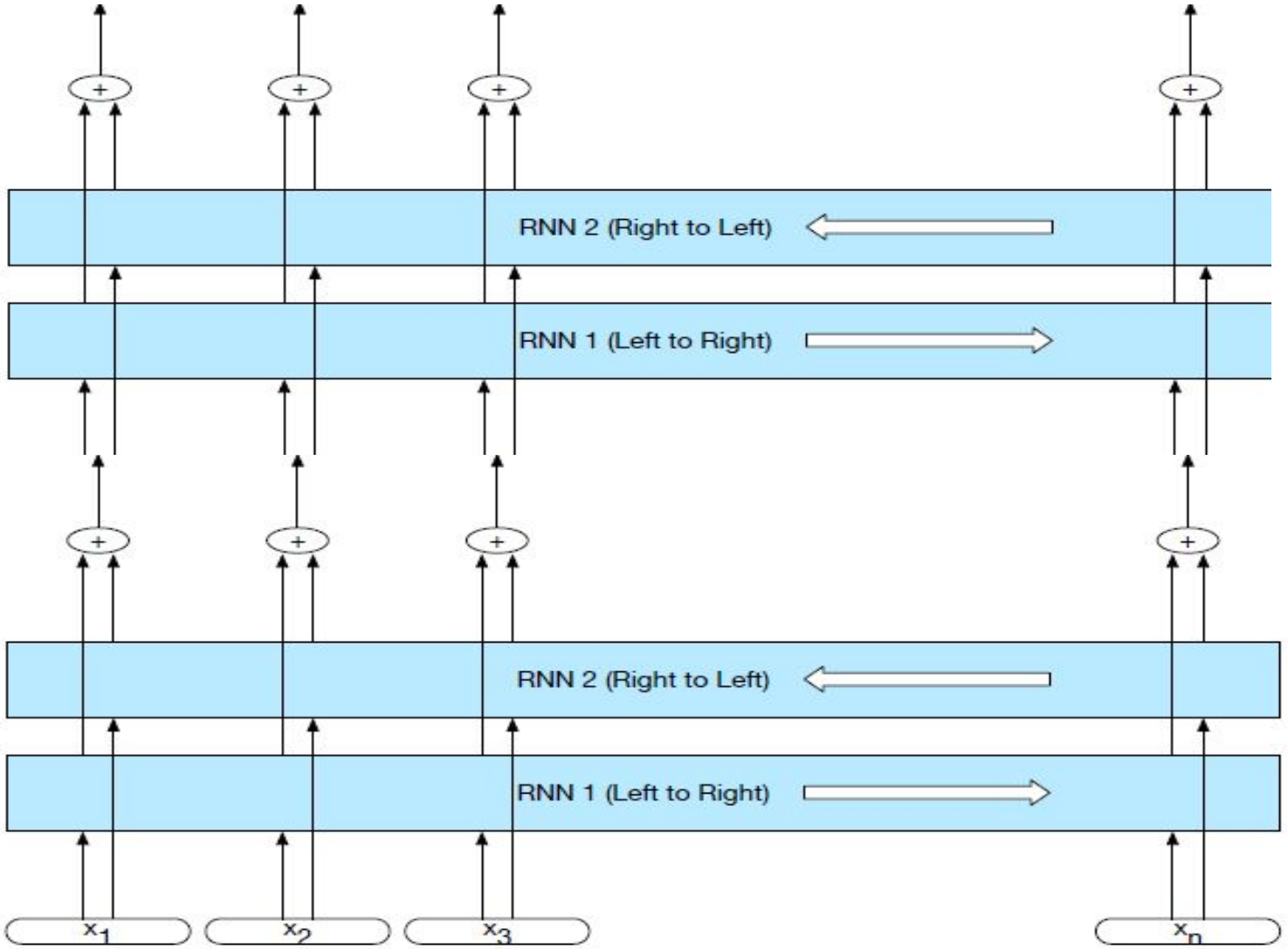
1. **Encoder**: принимает последовательность входов, $\mathbf{x}_{1:n}$ и создает на его основе последовательность контекстов, $\mathbf{h}_{1:n}$
2. **Контекстный вектор c**: Функция от $\mathbf{h}_{1:n}$ КОторая передает контекст Декодеру
3. **Decoder**: принимает контекст **c** на вход и генерирует последовательность скрытых состояний $\mathbf{h}_{1:m}$ из которых мы полчаем последовательность $\mathbf{y}_{1:m}$



Варианты архитектуры: Encoder

stacked Bi-LSTMs

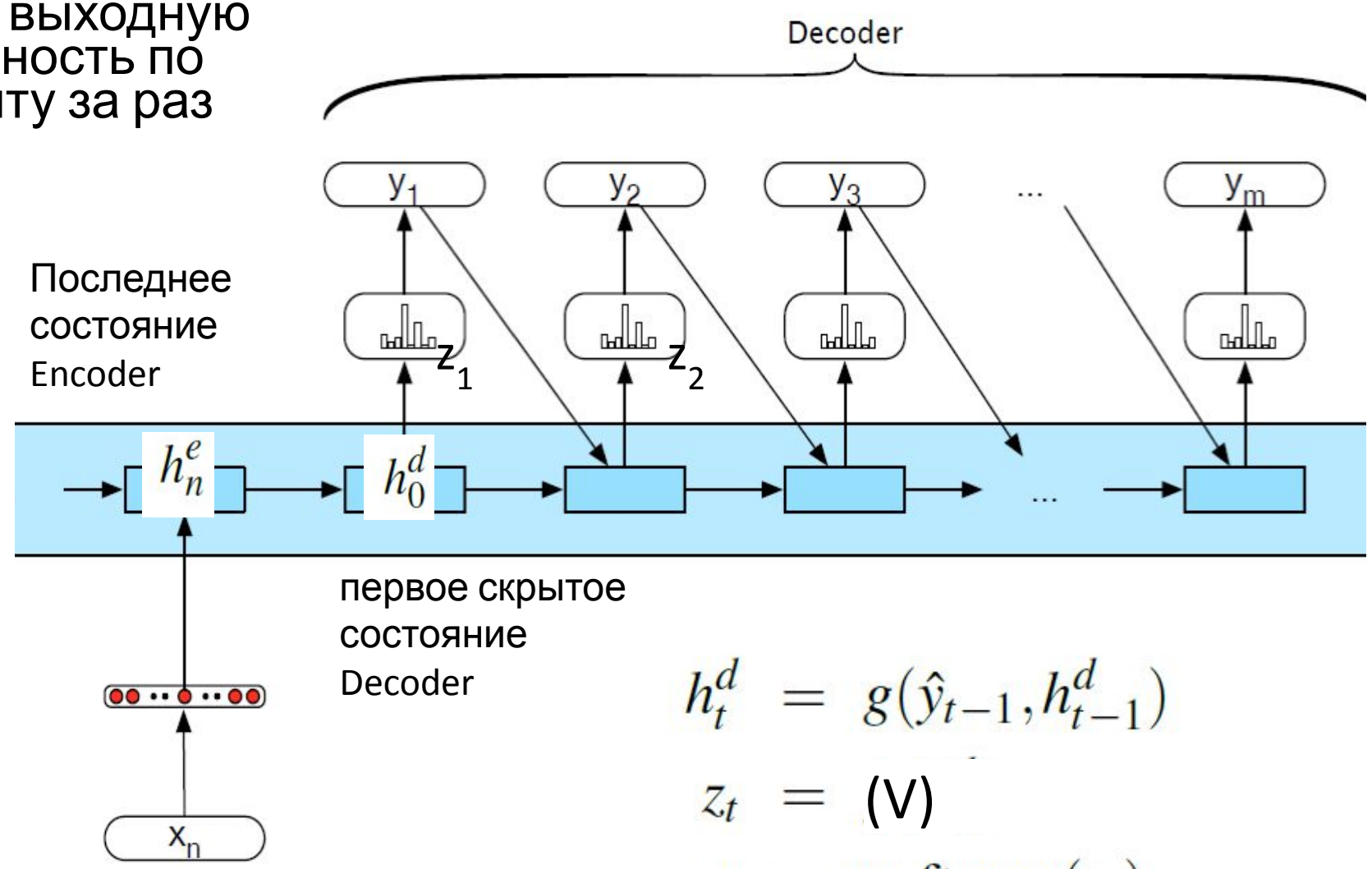
- Контекстуализированные представления для каждого временного шага: скрытые состояния от верхних слоев от прямого и обратного проходов



Decoder Основы

воспроизводит выходную последовательность по одному элементу за раз

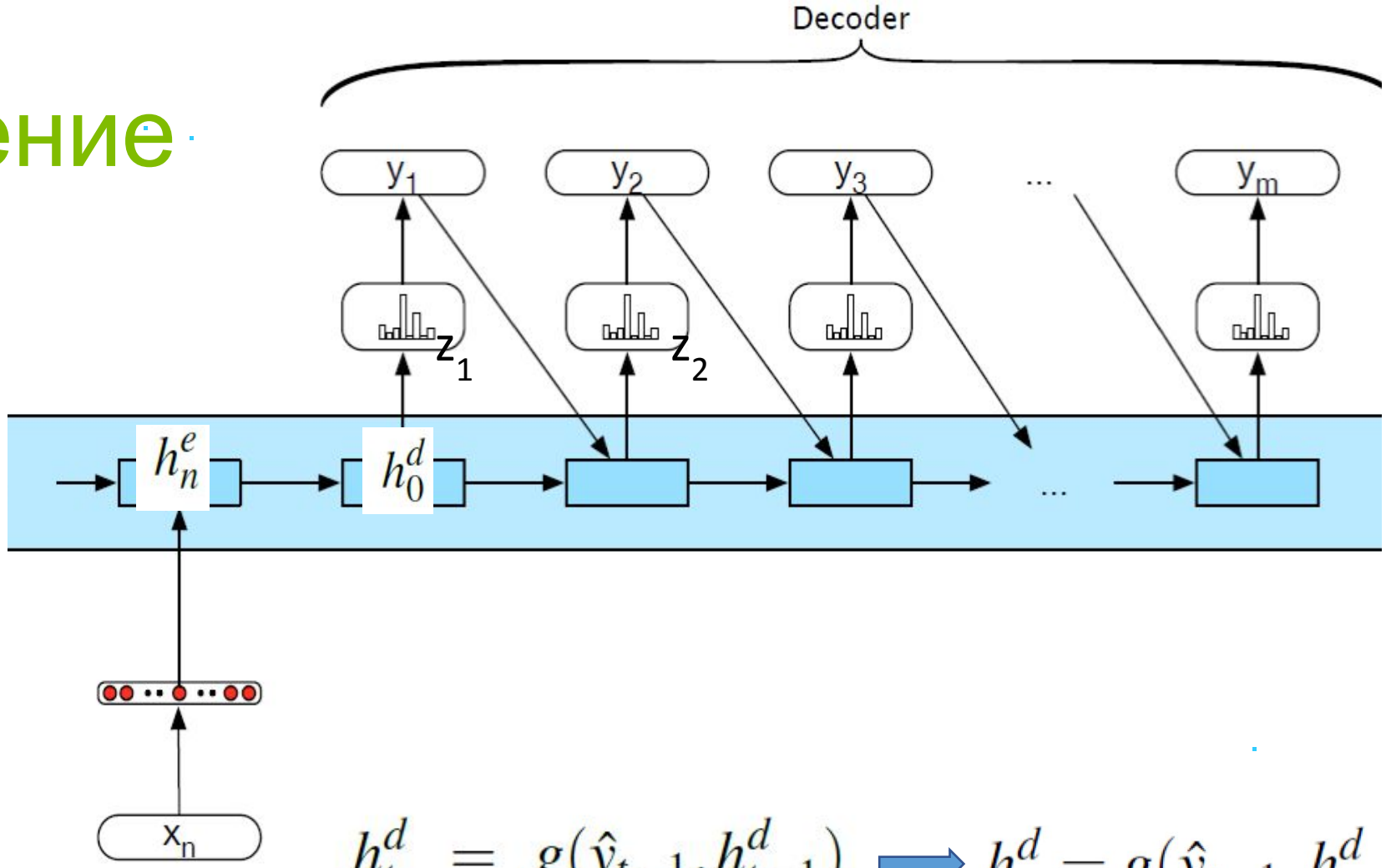
$$c = h_n^e$$
$$h_0^d = c$$



$$h_t^d = g(\hat{y}_{t-1}, h_{t-1}^d)$$
$$z_t = (V)$$
$$y_t = \text{softmax}(z_t)$$

Decoder Расширение

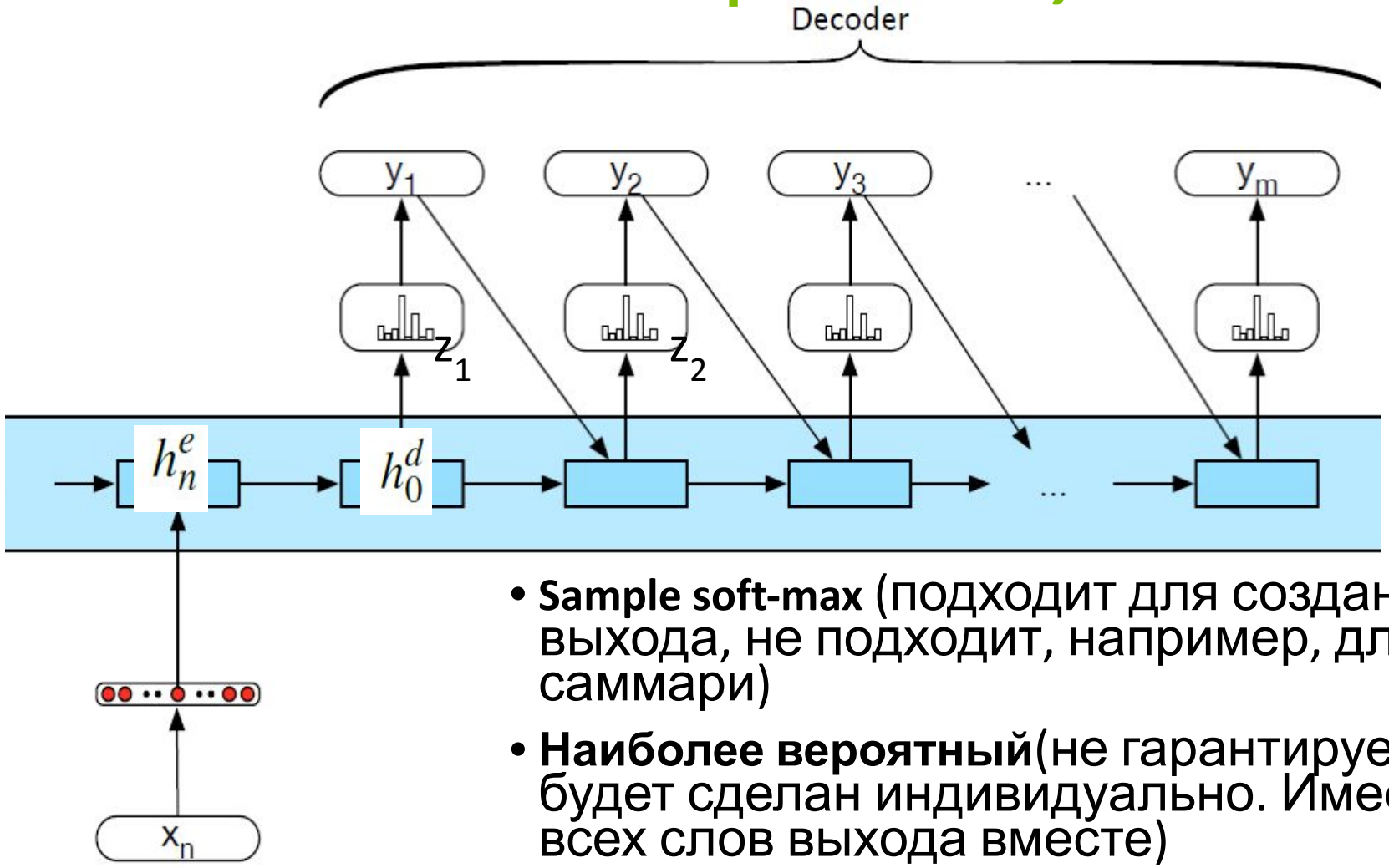
$$c = h_n^e$$
$$h_0^d = c$$



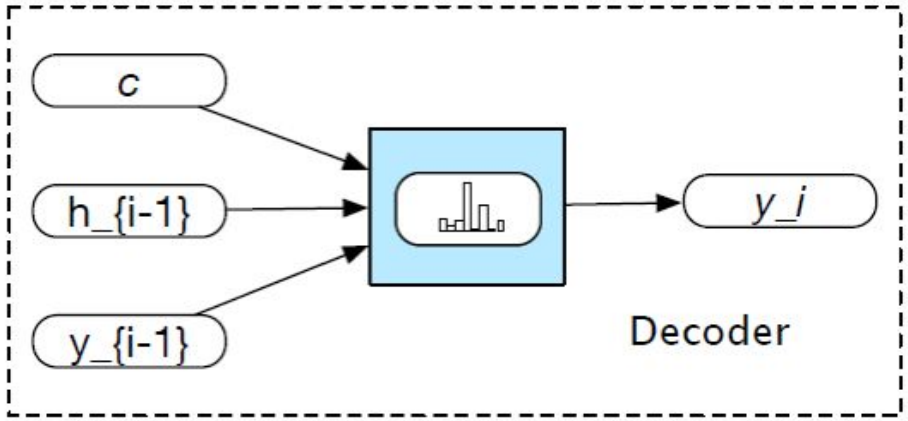
$$h_t^d = g(\hat{y}_{t-1}, h_{t-1}^d) \Rightarrow h_t^d = g(\hat{y}_{t-1}, h_{t-1}^d, c)$$
$$z_t = f(h_t^d)$$
$$y_t = \text{softmax}(z_t)$$

Context доступен на каждом шаге

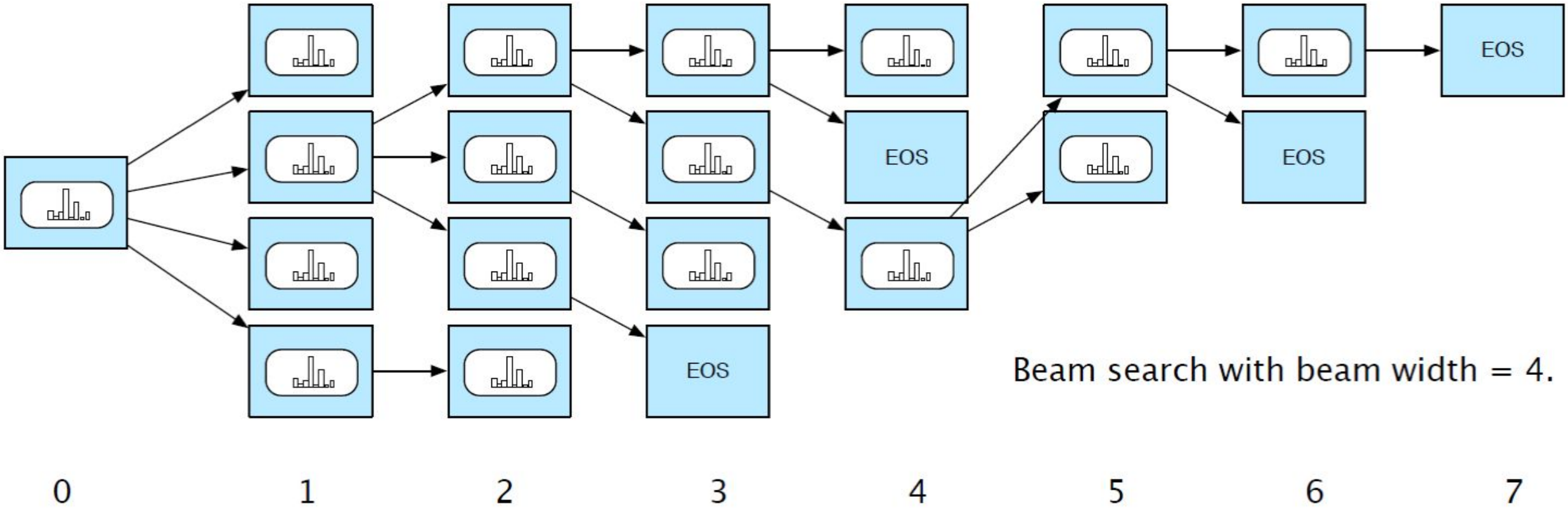
Decoder: Как выбирается *y?*



- **Sample soft-max** (подходит для создания одного выхода, не подходит, например, для перевода или саммари)
- **Наиболее вероятный**(не гарантирует, что выбор будет сделан индивидуально. Имеет смысл для всех слов выхода вместе)



- 4 наиболее вероятных «слова», декодированных из исходного состояния
- Подает каждый из них в декодере и хранит, 4 н**аиболее веротяные** последовательности из двух слов
- Подает самое последнее слово в декодер и сохраняет, 4 наиболее веротяныепоследовательности из трех слов …….
- Когда генерируется EOS. Остановить последовательность и уменьшить Beam на 1
- ….



План

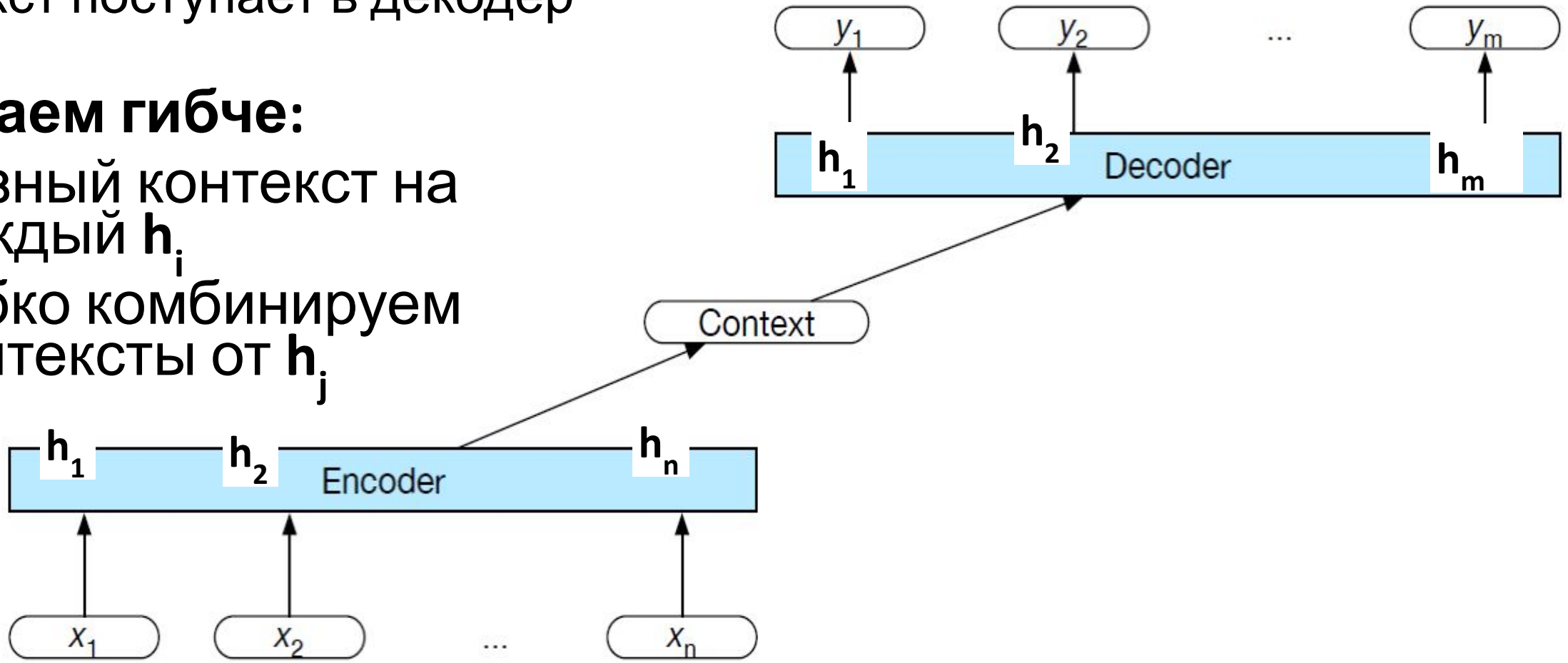
- Encoder Decoder
- **Attention**
- Self-attention

Переменный контекст: Attention

Контекстный вектор c:
функция от $\mathbf{h}_{1:n}$ через которую контекст поступает в декодер

Делаем гибче:

- Разный контекст на каждый $\mathbf{h}_i$
- Гибко комбинируем контексты от $\mathbf{h}_j$



Attention (1): динамически производный контекст

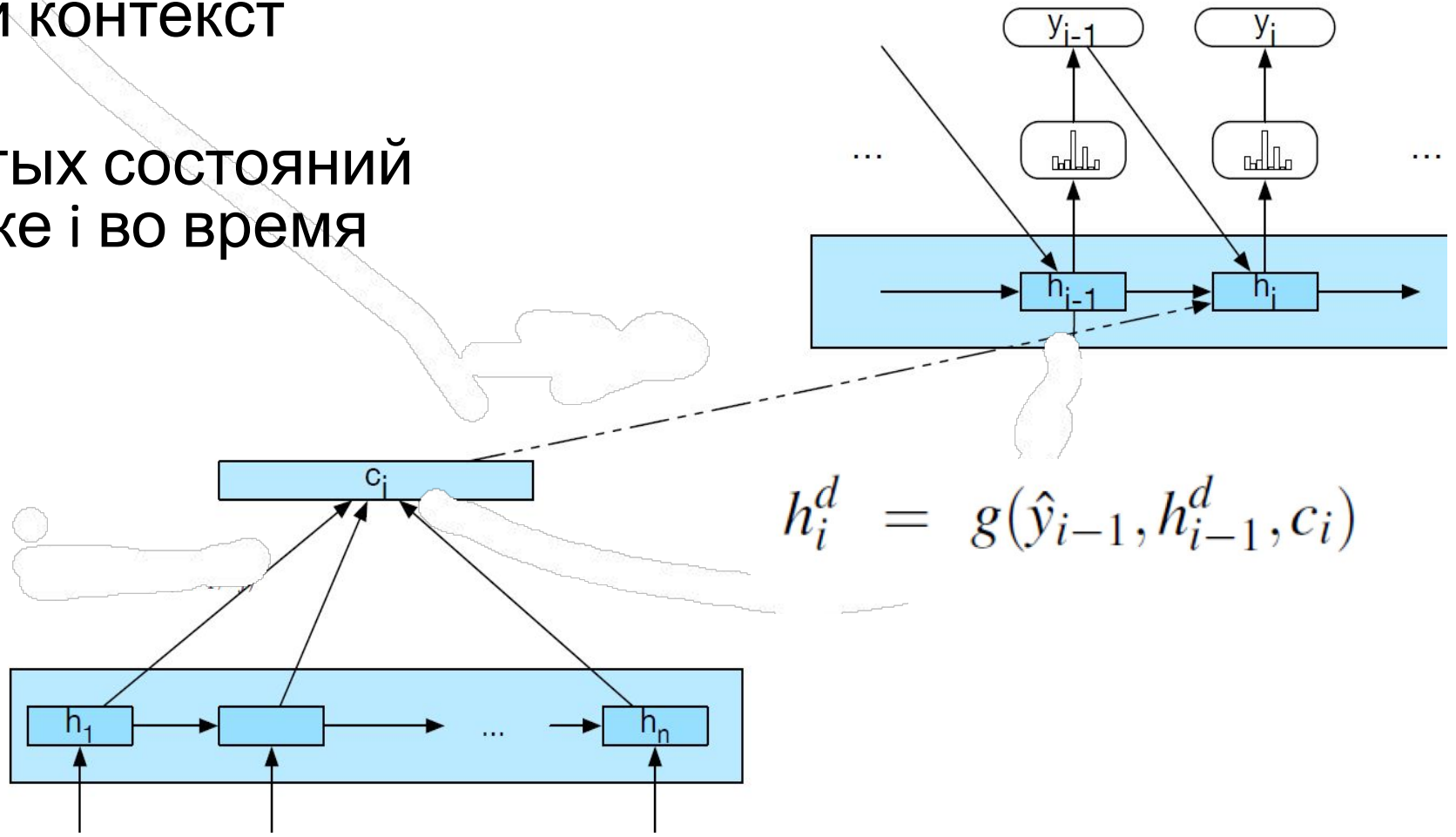
- Заменяем статичный контекст динамическим $\mathbf{c_i}$
- полученный из скрытых состояний encoder в каждой точке i во время декодирования

Варианты:

- Линейная комбинация

$$c_i = \sum_j \alpha_{ij} h_j^e$$

- α_{ij} Должен зависеть от



$$h_i^d = g(\hat{y}_{i-1}, h_{i-1}^d, c_i)$$

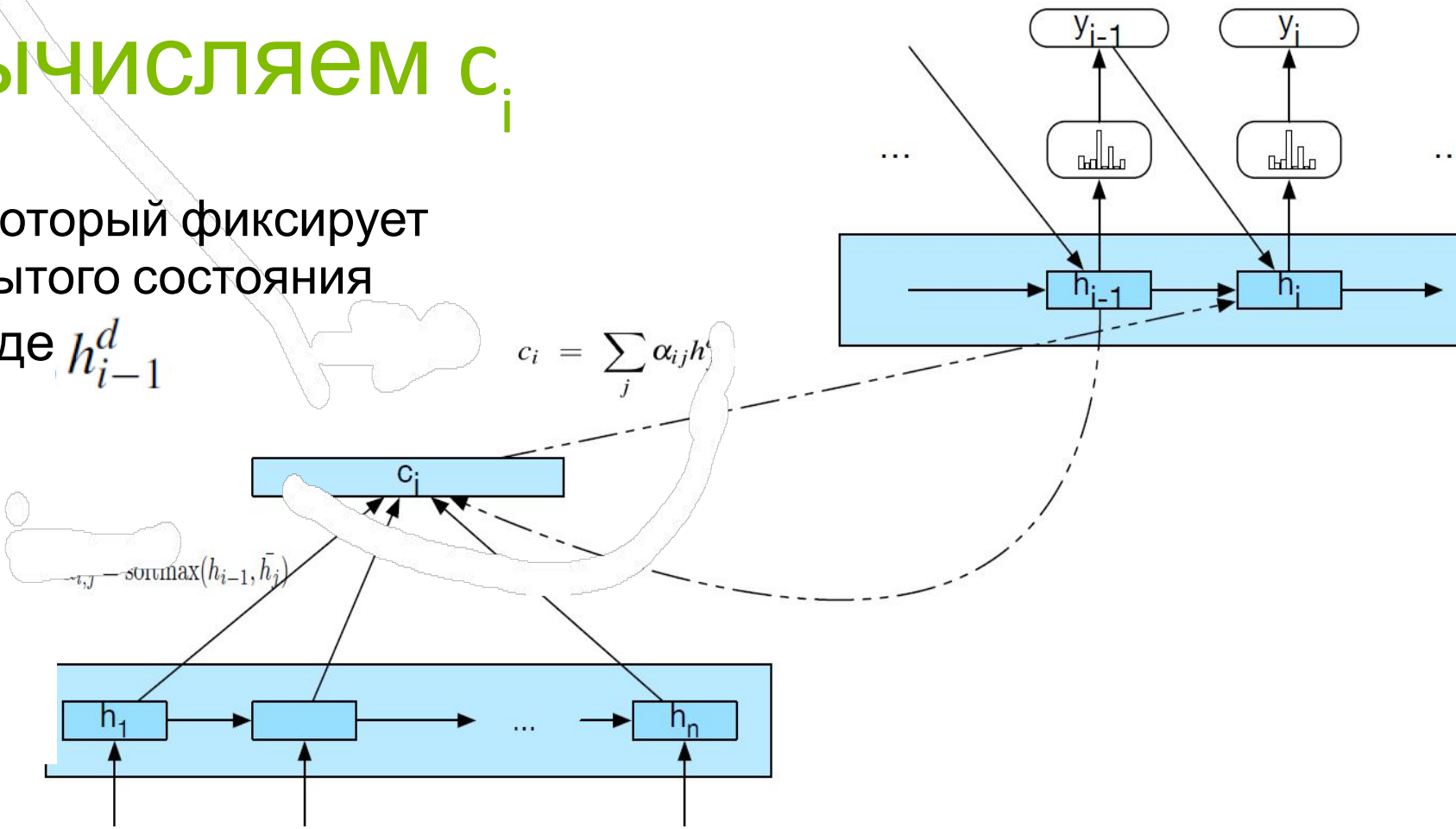
Attention (2): вычисляем c_i

- Вычислить вектор оценок, который фиксирует релевантность каждого скрытого состояния кодера для состояния декоде h_{i-1}^d

$$score(h_{i-1}^d, h_j^e)$$

- Простое сходство

$$score(h_{i-1}^d, h_j^e) = h_{i-1}^d \cdot h_j^e$$



- Предоставляет сети возможность узнать, какие аспекты сходства между состояниями decoder и encoder важны для текущего приложения.

$$score(h_{i-1}^d, h_j^e) = h_{t-1}^d W_s h_j^e$$

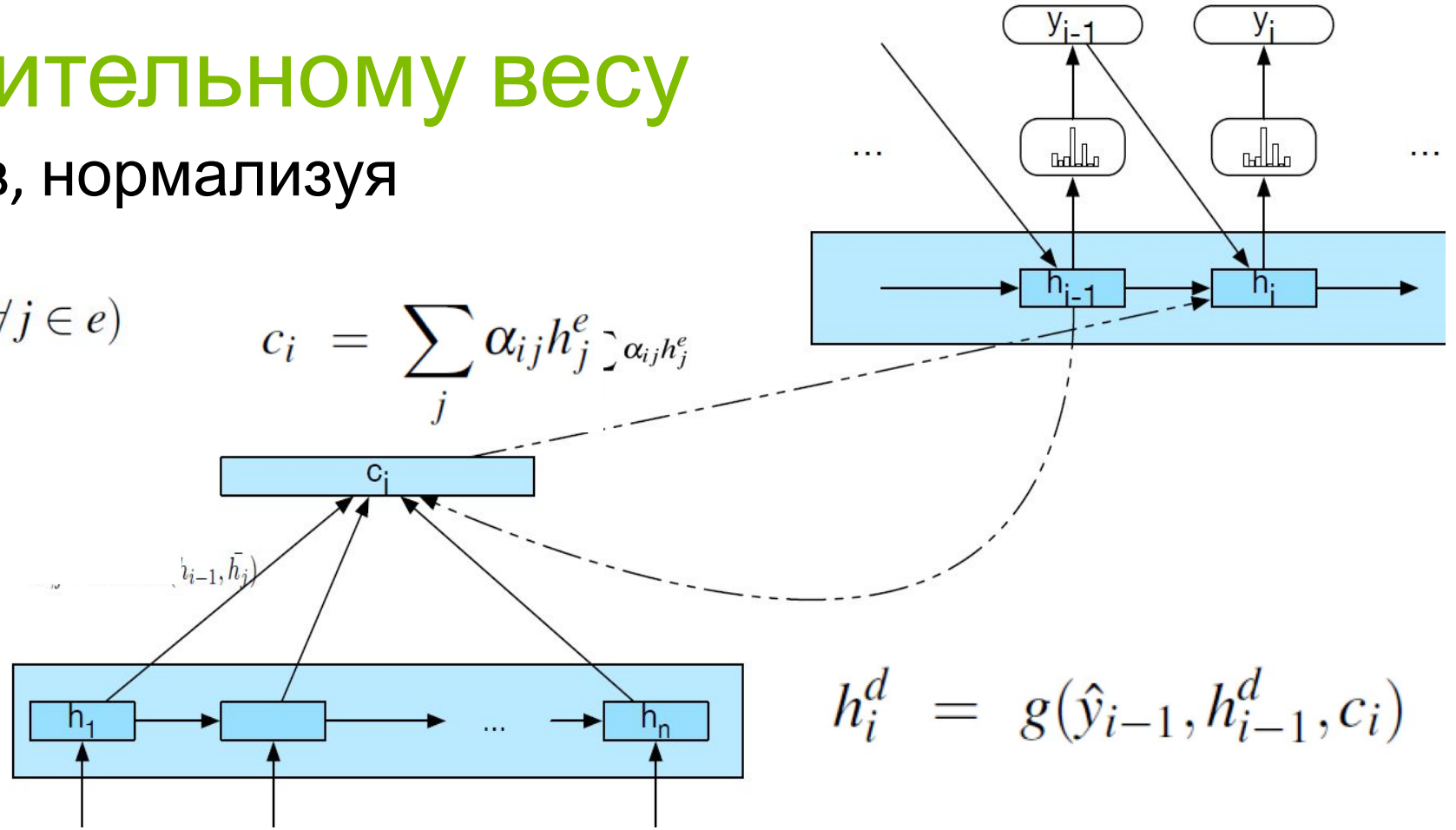
Attention (3): c_i от score к относительному весу

- Создаем вектор весов, нормализуя score

$$\alpha_{ij} = \text{softmax}(score(h^d_{i-1}, h^e_j) \ \forall j \in e)$$

$$= \frac{exp(score(h^d_{i-1}, h^e_j)}{\sum_k exp(score(h^d_{i-1}, h^e_k))}$$

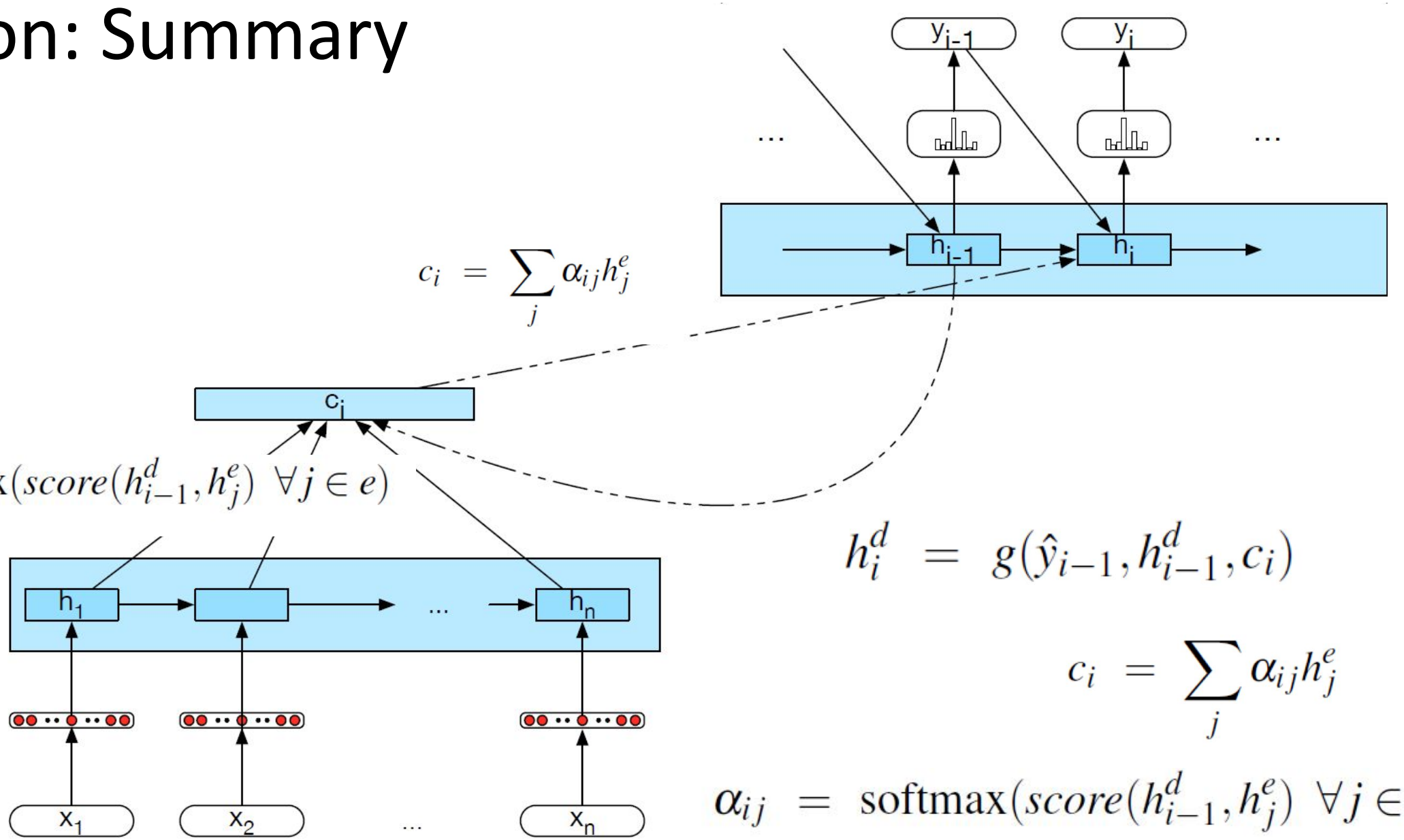
$$c_i = \sum_j \alpha_{ij} h^e_j$$



$$h^d_i = g(\hat{y}_{i-1}, h^d_{i-1}, c_i)$$

Цель достигнута: вычислить вектор контекста фиксированной длины для текущего состояния декодера, взяв средневзвешенное значение по всем скрытым состояниям кодера.

Attention: Summary



$$c_i = \sum_j \alpha_{ij} h_j^e$$

$$\alpha_{ij} = \text{softmax}(score(h_{i-1}^d, h_j^e) \ \forall j \in e)$$

$$h_i^d = g(\hat{y}_{i-1}, h_{i-1}^d, c_i)$$

$$c_i = \sum_j \alpha_{ij} h_j^e$$

$$\alpha_{ij} = \text{softmax}(score(h_{i-1}^d, h_j^e) \ \forall j \in e)$$

Attention: Summary2