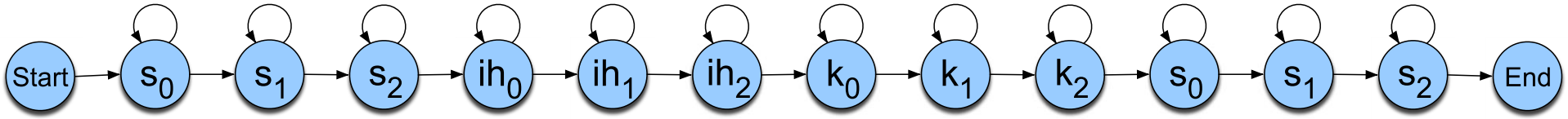
Распознавание речи

Современные подходы

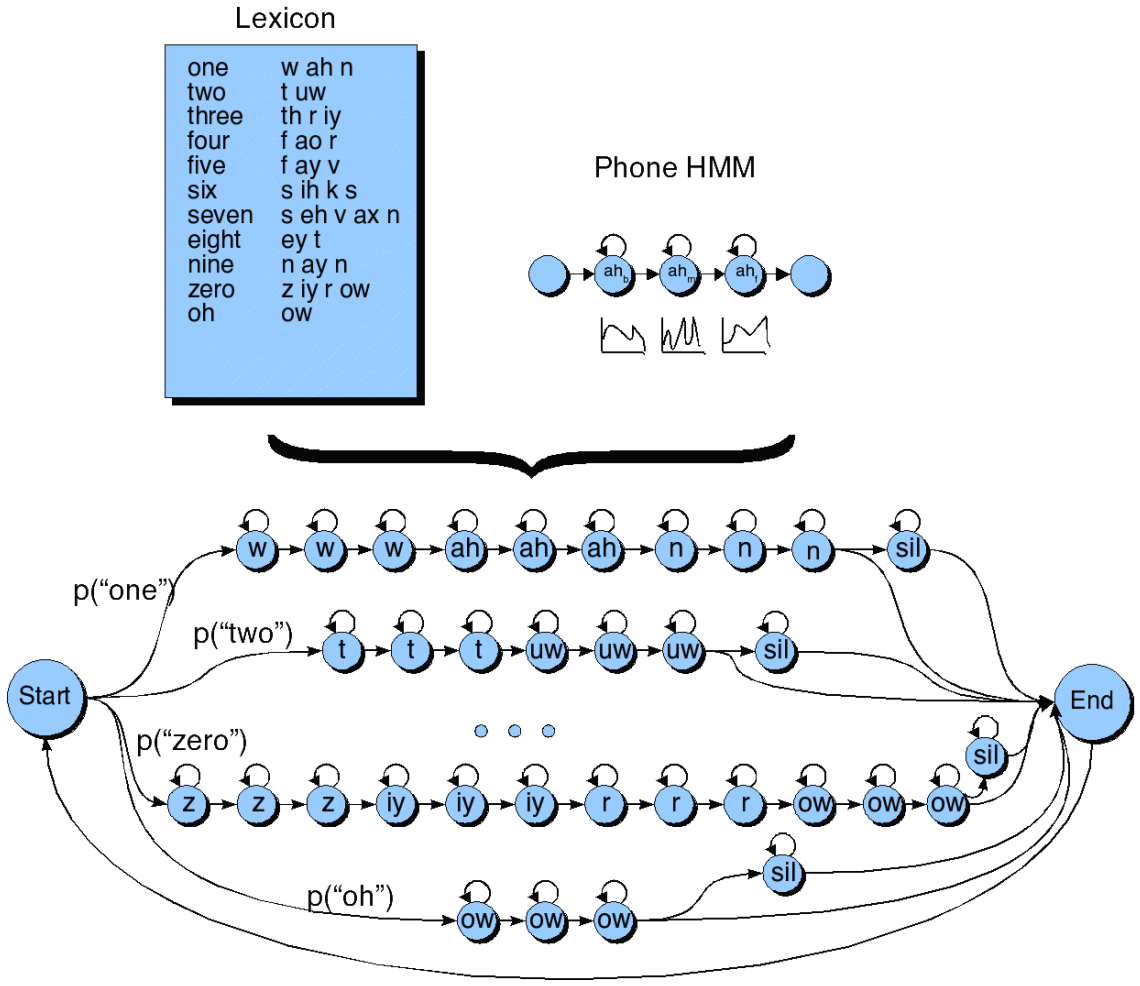
Декодирование для чисел

- HMM для одного слова



  - $\mathrm{Q} = q_1, \ldots, q_N = N$ – состояний модели – это наши субфонемы
  - $\mathrm{A} = a_{11}, a_{12}, a_{1n}, \ldots, a_{nn}$, - матрица вероятностей перехода. a_{ij} - вероятность перехода между субфонемами или переход сам на себя. Для каждого слова создается граф состояний HMM.
  - $\mathrm{B} = b_j(o_t),$ – набор наблюдений. Каждый элемент элемент выражает вероятность пронаблюдать вектор признаков, который генерируется состоянием j.

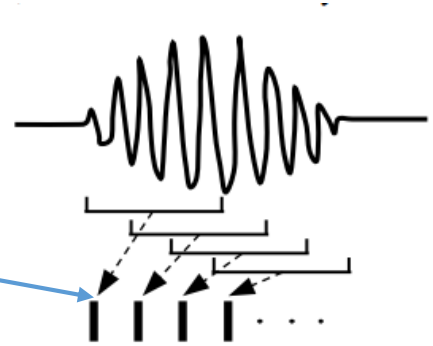
Декодирование для изолированных слов



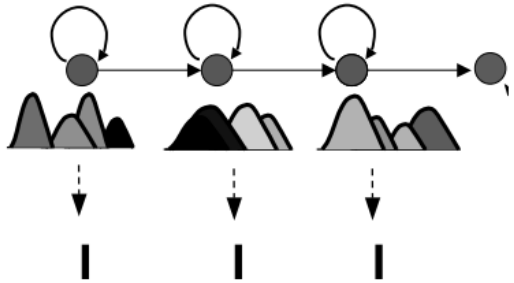
- Матрицы *A* и *B* получаем в результате обучения модели алгоритмом “Вперед-назад”.

Декодирование для изолированных слов

- $\widehat{W} = \underset{W \epsilon \mathcal{L}}{\operatorname{argmax}} P(O|W)P(W)$

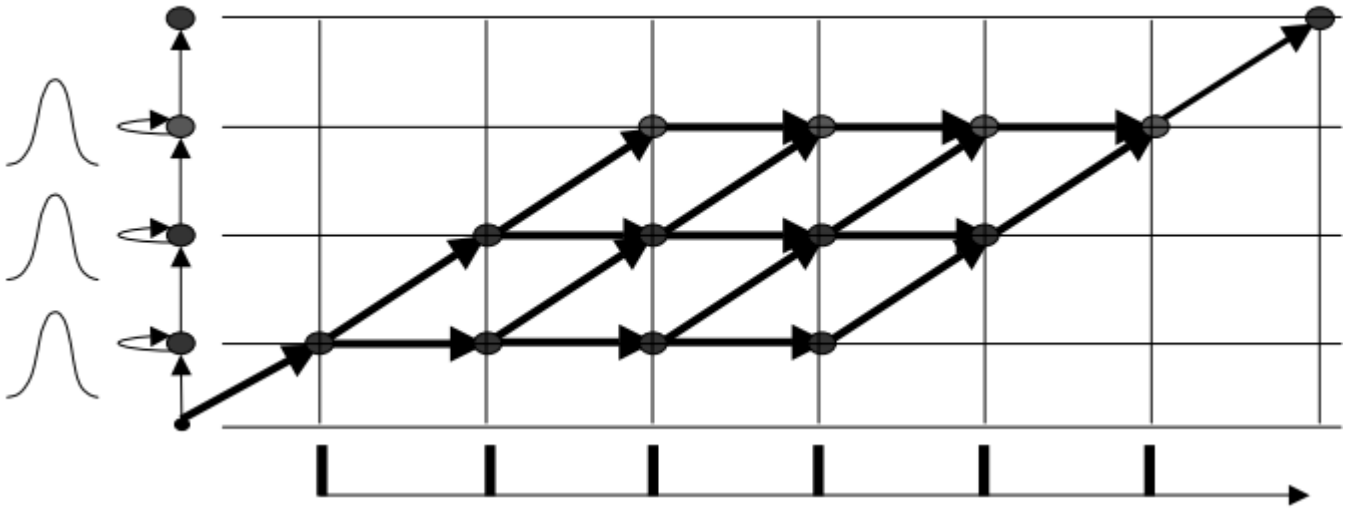


- $P(W)$ - априорная вероятность слова
- $P(O|W)$ - вероятность наблюдения признаков сгенерированных словом W
- Мы представляем каждое слово с помощью HMM
  - HMM - это графическая форма функции плотности вероятности для изменяющихся во времени данных



$P(O|W)$- для изолированного слова

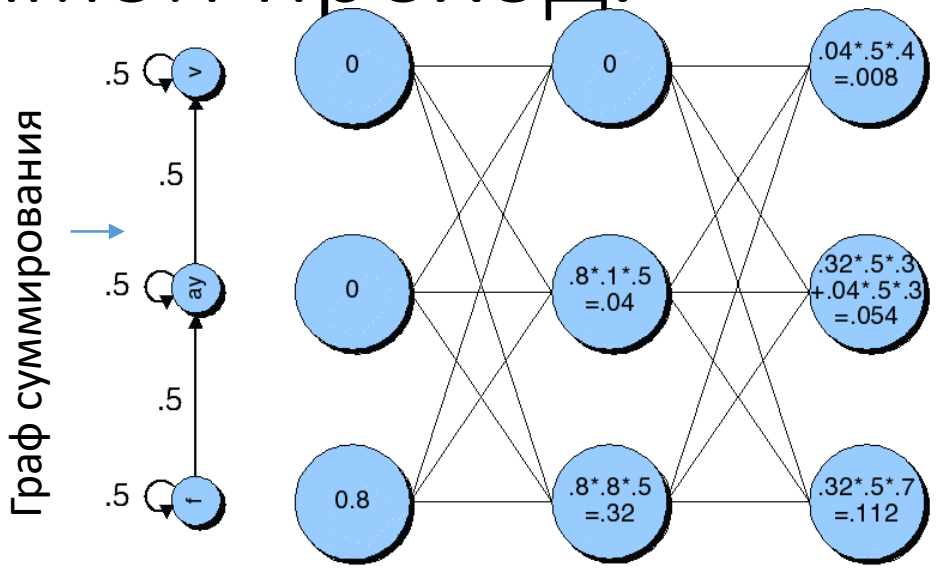
- *Реальное последовательность состояний, которое генерирует O неизвестно*
  - $P(O|W)$ - должно рассматривать все возможные состояния
    - $P(O|W) = \sum_{q \in \{\text{все состояни}\}} P(O, q|\, W)$



Вероятность последовательности это сумма вероятностей по всем последовательностям

$P(O|W)$ - для слова “five”. Сумма по всем возможным путям. Прямой проход.

- f ay ay ay ay v v v v
- f f ay ay ay ay v v v
- f f f f ay ay ay ay v
- f f ay ay ay ay ay ay v
- f f ay ay ay ay ay ay ay ay v
- f f ay v v v v v v v



наблюдения →

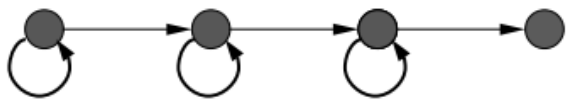
V	0	0	0.008	0.0093	0.0114	0.00703	0.00345	0.00306	0.00206	0.00117
AY	0	0.04	0.054	0.0664	0.0355	0.016	0.00676	0.00208	0.000532	0.000109
F	0.8	0.32	0.112	0.0224	0.00448	0.000896	0.000179	4.48e-05	1.12e-05	2.8e-06
Time	1	2	3	4	5	6	7	8	9	10
B	*f* 0.8	*f* 0.8	*f* 0.7	*f* 0.4	*f* 0.4	*f* 0.4	*f* 0.4	*f* 0.5	*f* 0.5	*f* 0.5
	ay 0.1	*ay* 0.1	*ay* 0.3	*ay* 0.8	*ay* 0.8	*ay* 0.8	*ay* 0.8	*ay* 0.6	*ay* 0.5	*ay* 0.4
	v 0.6	*v* 0.6	*v* 0.4	*v* 0.3	*v* 0.3	*v* 0.3	*v* 0.3	*v* 0.6	*v* 0.8	*v* 0.9
	p 0.4	*p* 0.4	*p* 0.2	*p* 0.1	*p* 0.1	*p* 0.1	*p* 0.1	*p* 0.1	*p* 0.3	*p* 0.3
	iy 0.1	*iy* 0.1	*iy* 0.3	*iy* 0.6	*iy* 0.6	*iy* 0.6	*iy* 0.6	*iy* 0.5	*iy* 0.5	*iy* 0.4

Классификация изолированных слов

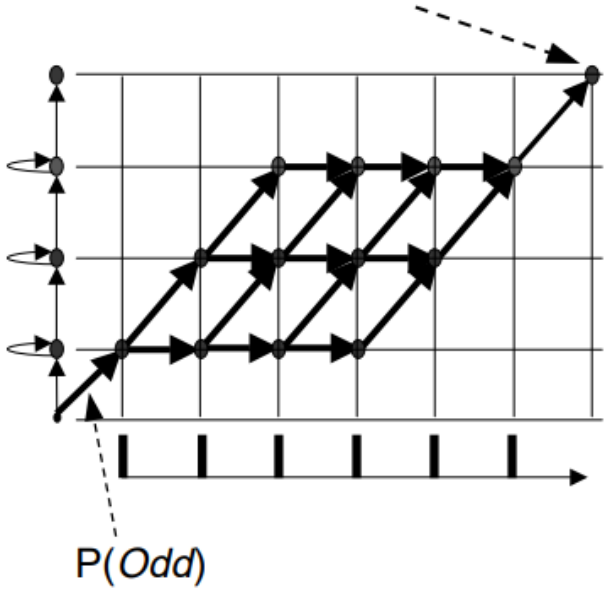
Выбираем:

$$\underset{W \in \mathcal{L}}{\operatorname{argmax}}\, P(O|W)P(W)$$

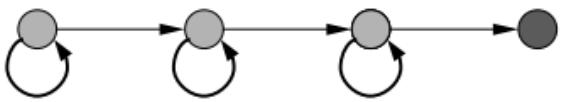
HMM для Odd



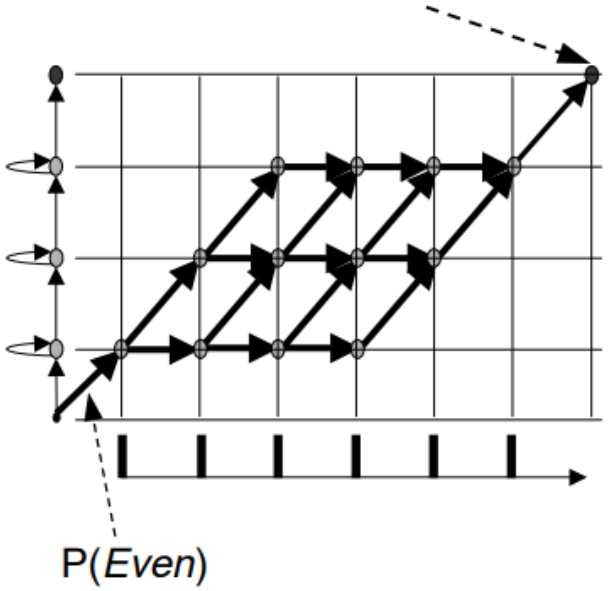
$P(Odd)P(O|Odd)$



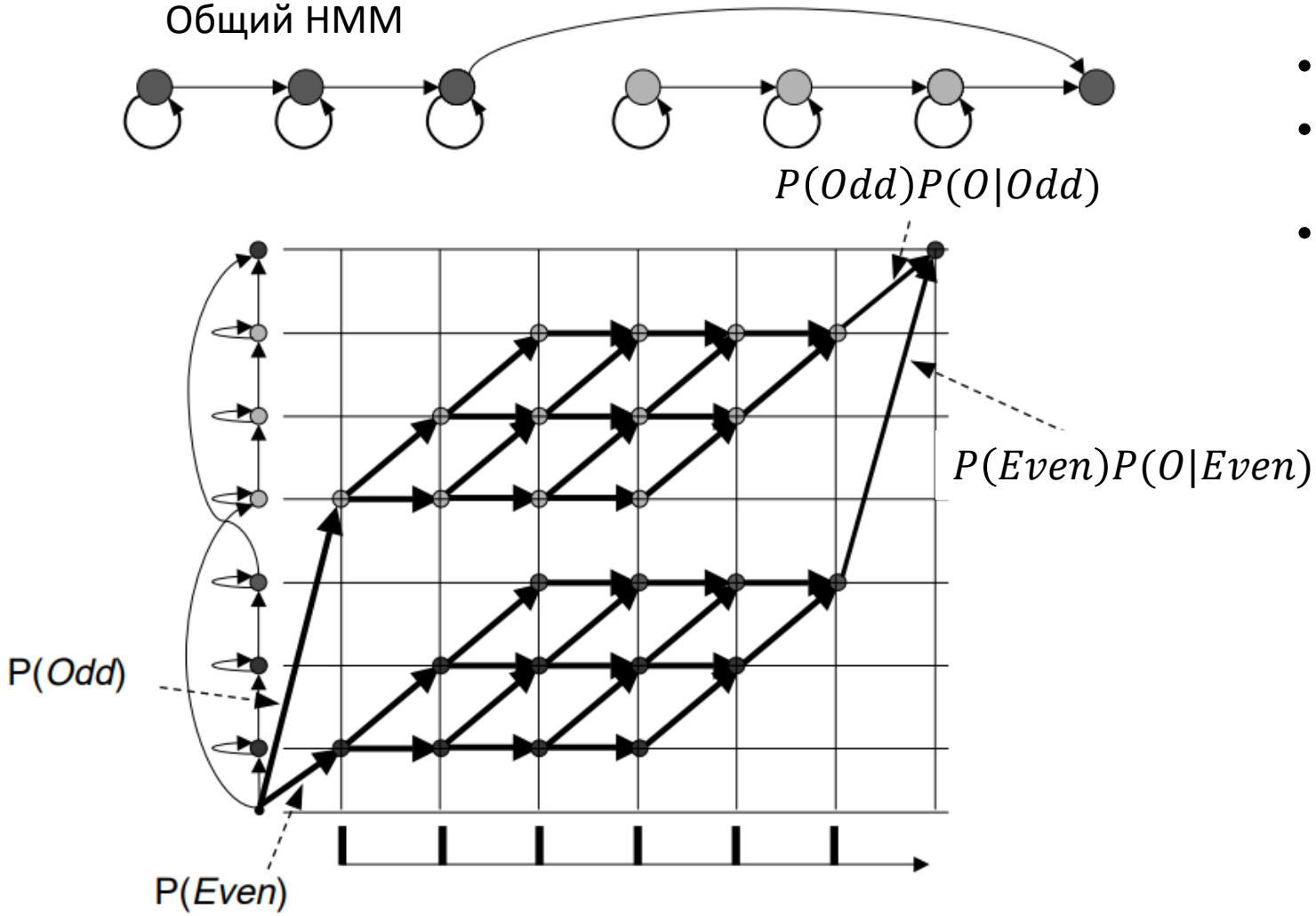
HMM для Even



$P(Even)P(O|Even)$



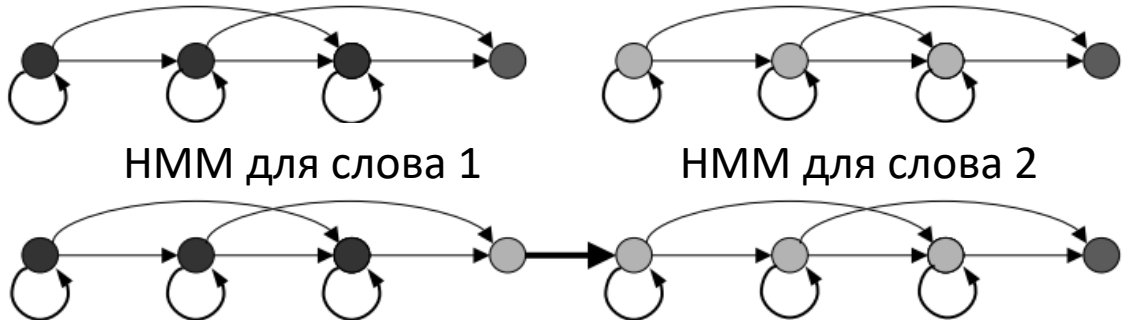
Классификация между Odd и Even



- Используем общий HMM
- Для одной фонемы используем одно состояние
- Для декодирования используем Viterbi.
  - $\max\{ P(Odd)P(O|Odd), P(Even)P(O|Even)\}$

HMM для последовательности слов

- Складываем HMM для каждого слова в последовательности

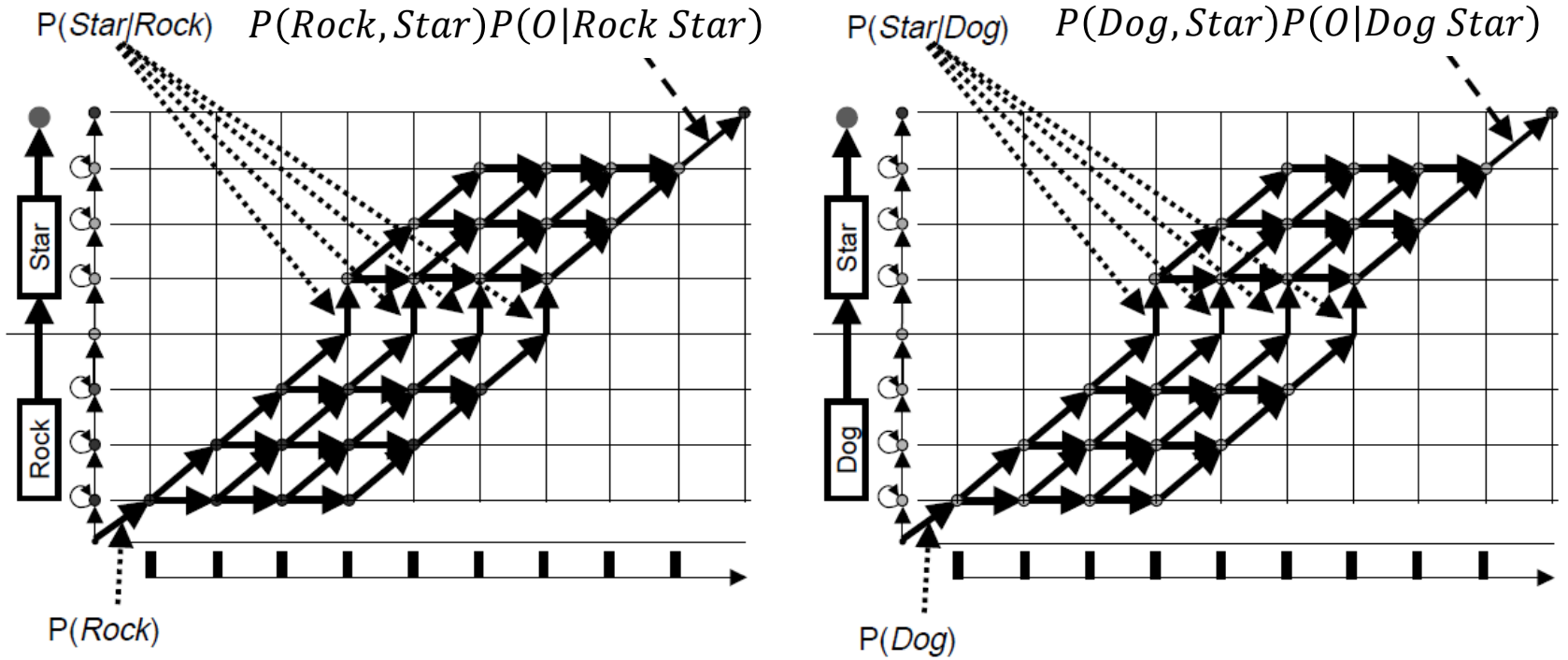


Комбинированный HMM для последовательности слово 1 – слово 2

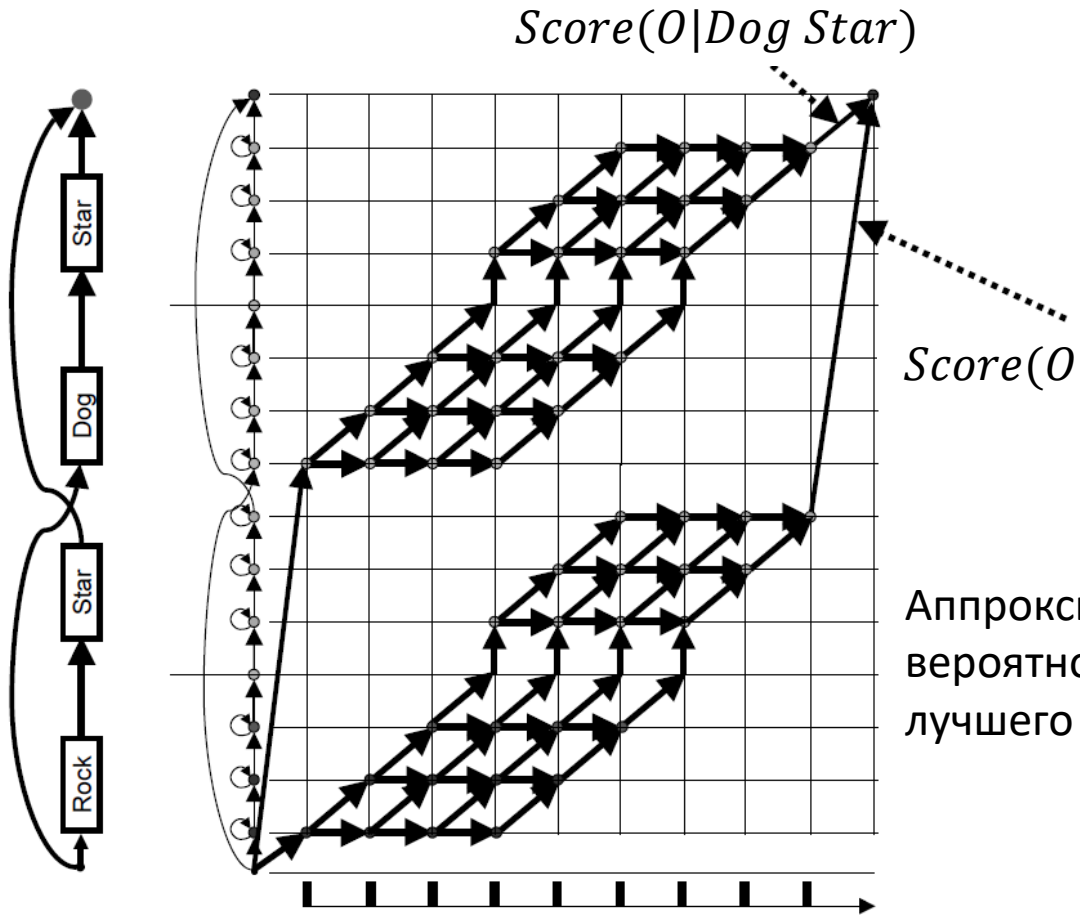
- HMM для слов - сама по себе конкатенация HMM для фонем
  - фонем намного меньше, чем слов, и встречаются чаще в данных обучения
  - слова, которые никогда не видели в данных обучения, могут быть построены из фонемных HMM

Классификация двух последовательностей

- Классифицируем “Rock Star” vs “Dog Star”
  - Прямой проход и выбираем максимум:
    - $argmax\ \{P(Rock, Star)P(O|Rock\ Star), P(Dog\ , Star)P(O|Dog\ Star)\}$



Декодирование последовательности слов

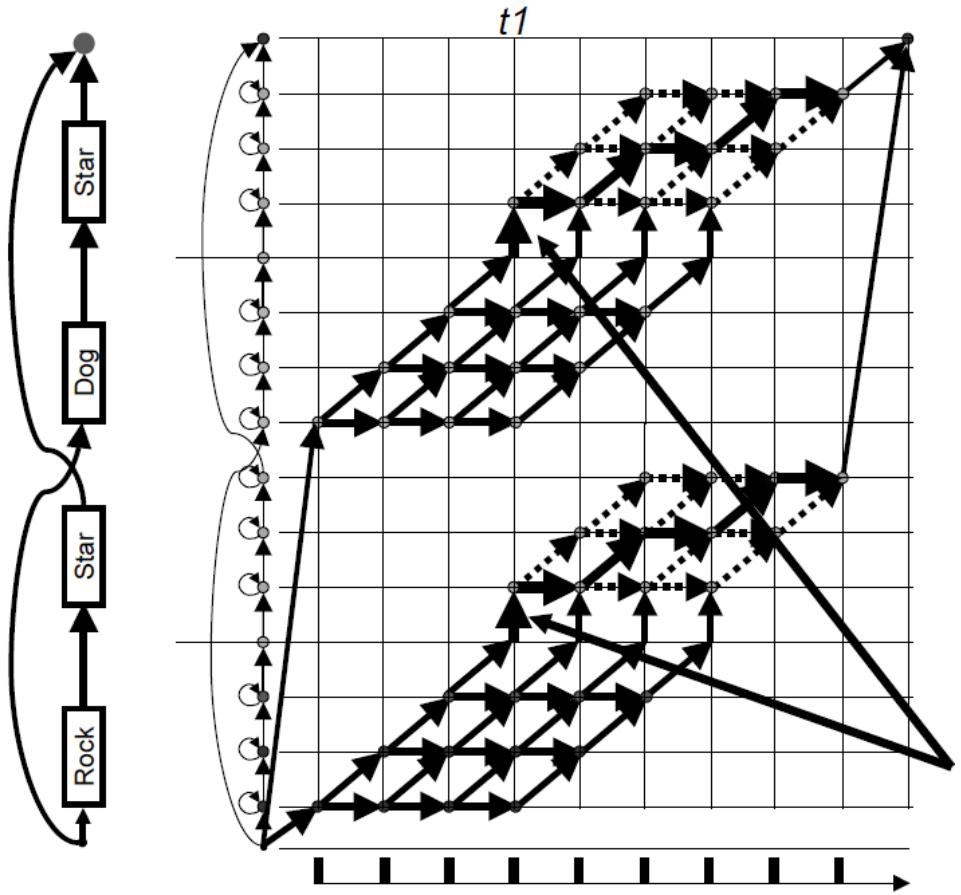


Используем Viterbi для получения скора каждого пути

$Viterbi = \max(\max(dogstar), \max(rockstar))$

Аппроксимируем общую вероятность выбором лучшего пути

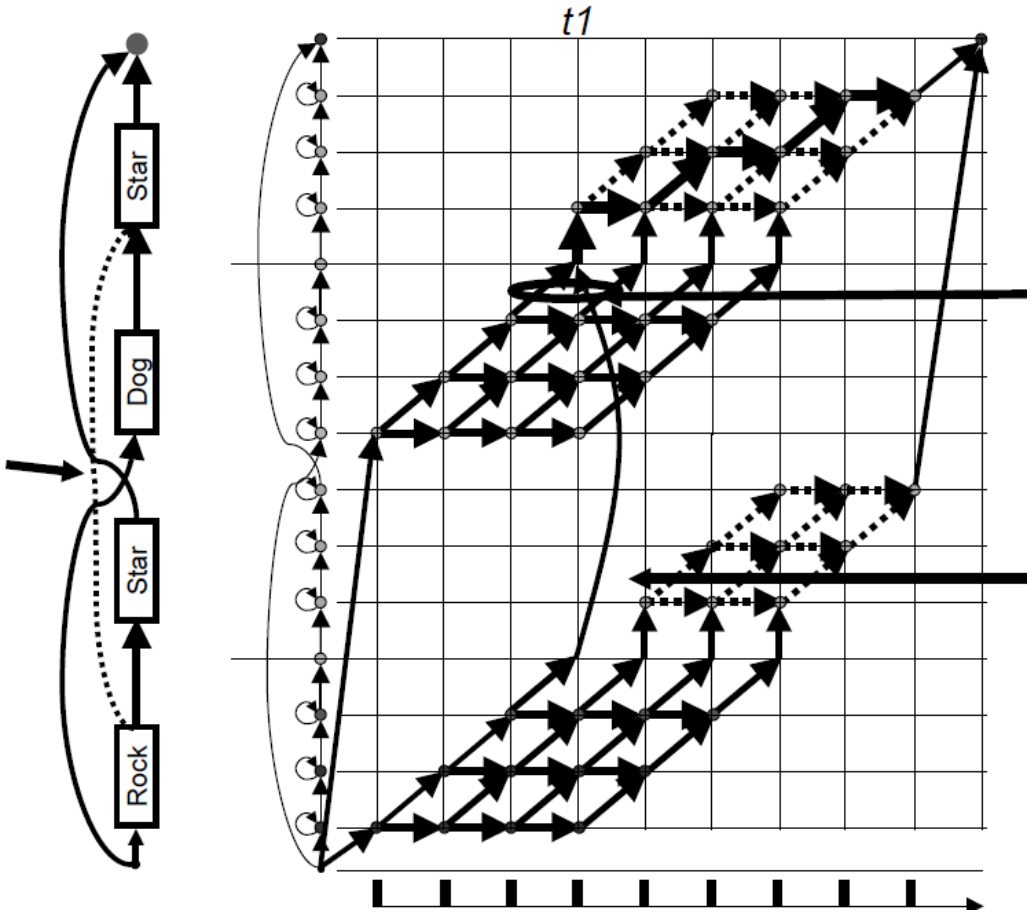
Декодирование последовательностей



Для финальной точки наилучший путь через “STAR” одинаковый для обоих решеток

Мы можем выбирать между “Dog” и “Rock” прямо здесь, потому что фичи этих путей идентичны.

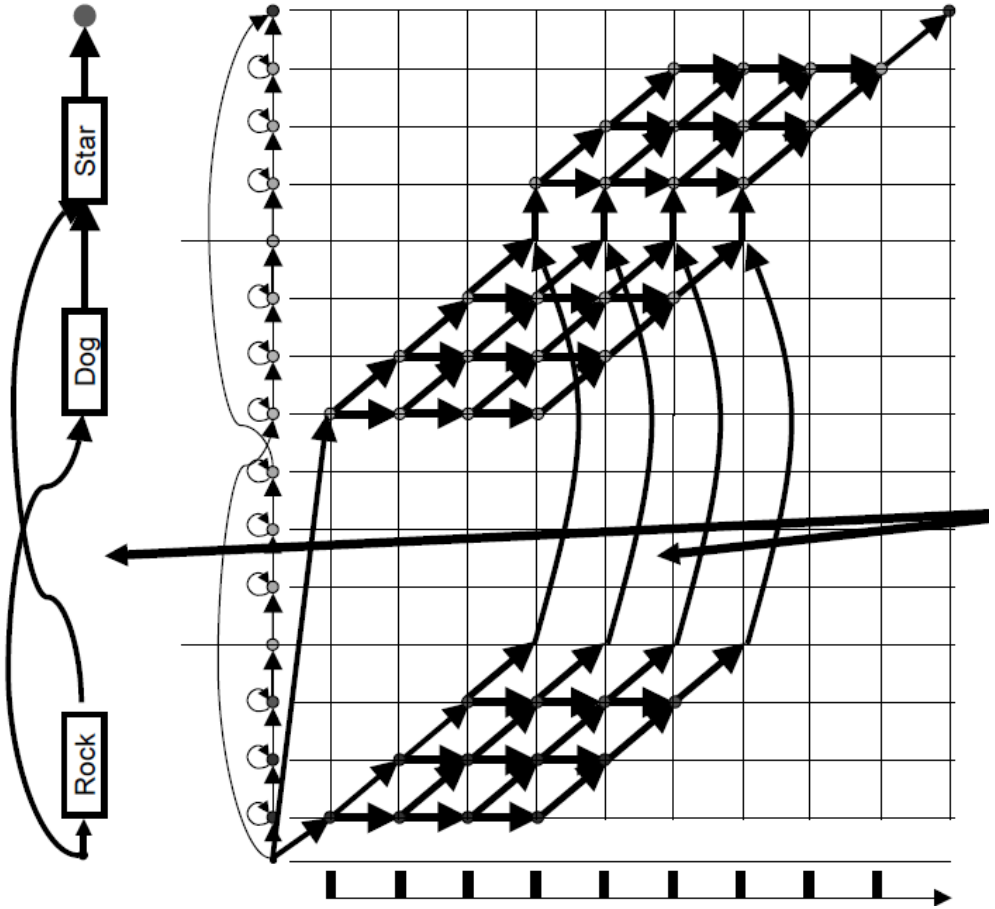
Декодирование последовательностей



Мы выбираем наилучший скор двух входящих ребер тут

Эту часть решетки удаляем

Декодирование последовательности



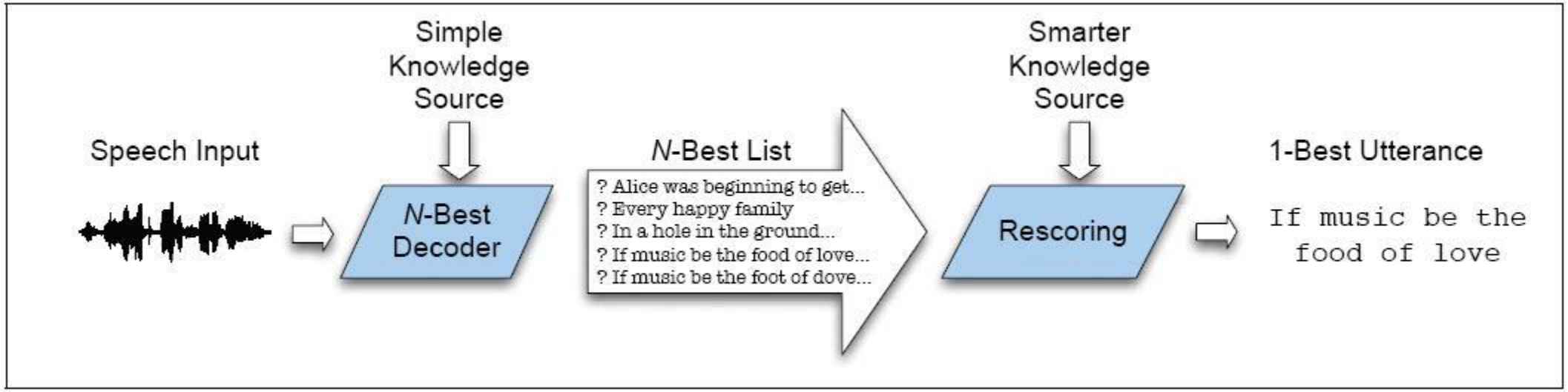
Мы теперь выполняем декодирование Витерби на этом свернутом графе.

Эту копию решетки для “STAR” полностью удаляем

Проблемы с Витерби

- Трудно интегрировать сложные источники знаний
  - Триграммные грамматики
  - Парсерное решение или нейронную сеть для языковой модели
    - длинные зависимости нарушают предположения динамического программирования
  - Знания, которые идут не справа налево
    - Следующие слова могут помочь предсказать предыдущие слова
- Решения
  - Возвращать множество гипотез и затем ранжировать их по дополнительным знаниям
  - Использовать другой поисковый алгоритм

Многопроходный поиск



Расширенные алгоритмы поиска, разобрать самостоятельно.

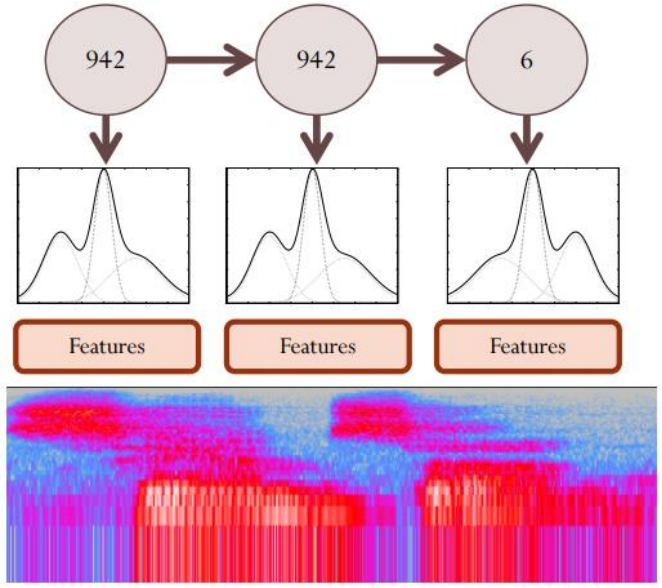
Нейросети для ASR

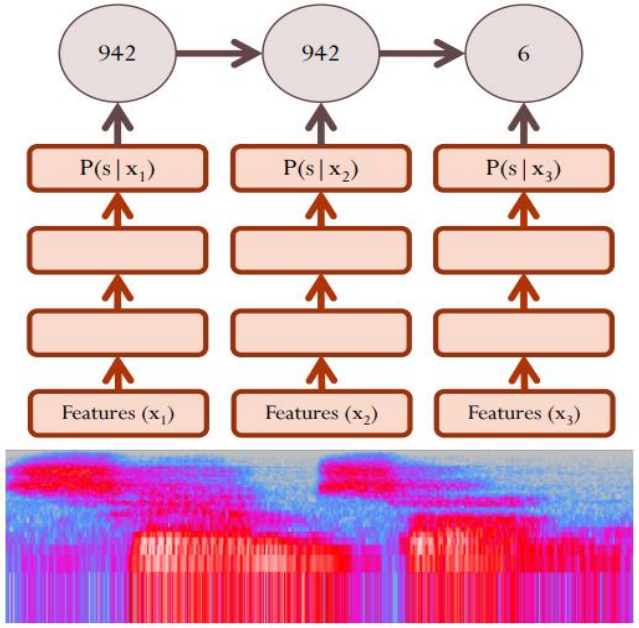
- Гибридная модель

Hidden Markov Model (HMM):

Акустическая модель:

Аудио вход:





$P(q|o)$ - классификатор состояний
$P(q)$ – априорное распределение состояний

GMM Акустическая модель
Моделирует: $P(o|q)$
o: признаки
q: состояние HMM

Гибридная DNN акустическая модель
Моделирует: $P(q|o)$,
нам нужна $P(o|q) = P(q|o)P(o)/P(q)$
$P(o) = const$ – в процессе декодирования
$P(o|q) = P(q|o)/P(q)$

Целевая функция обучения DNN

- Обучение с учителем, минимизируем ошибку классификации
- Обычно – кросс энтропия
  - $Loss(x, y; W, b) = -\sum_{k=1}^{K}(y = k)\log f\ (x)_k$
- Ошибка считается по каждому окну.
- Метки классов получаем из HMM-GMM (принудительное выравнивание)

Обучение DNN

- Сеть состоит из более чем одного слоя между входом и выходом
  - Слой состоит из скрытых элементов
  - $j^{\text{й}}$ элемент слоя использует нелинейную функцию активации, например, $y_j = \ logistic(x_j) = \frac{1}{1 + e^{-x_j}}$
  - Для маппирования его суммарного входа $x_j = b_j + \sum_i y_i w_{ij}$

Выход предыдущего слоя

смещение

Вес связи

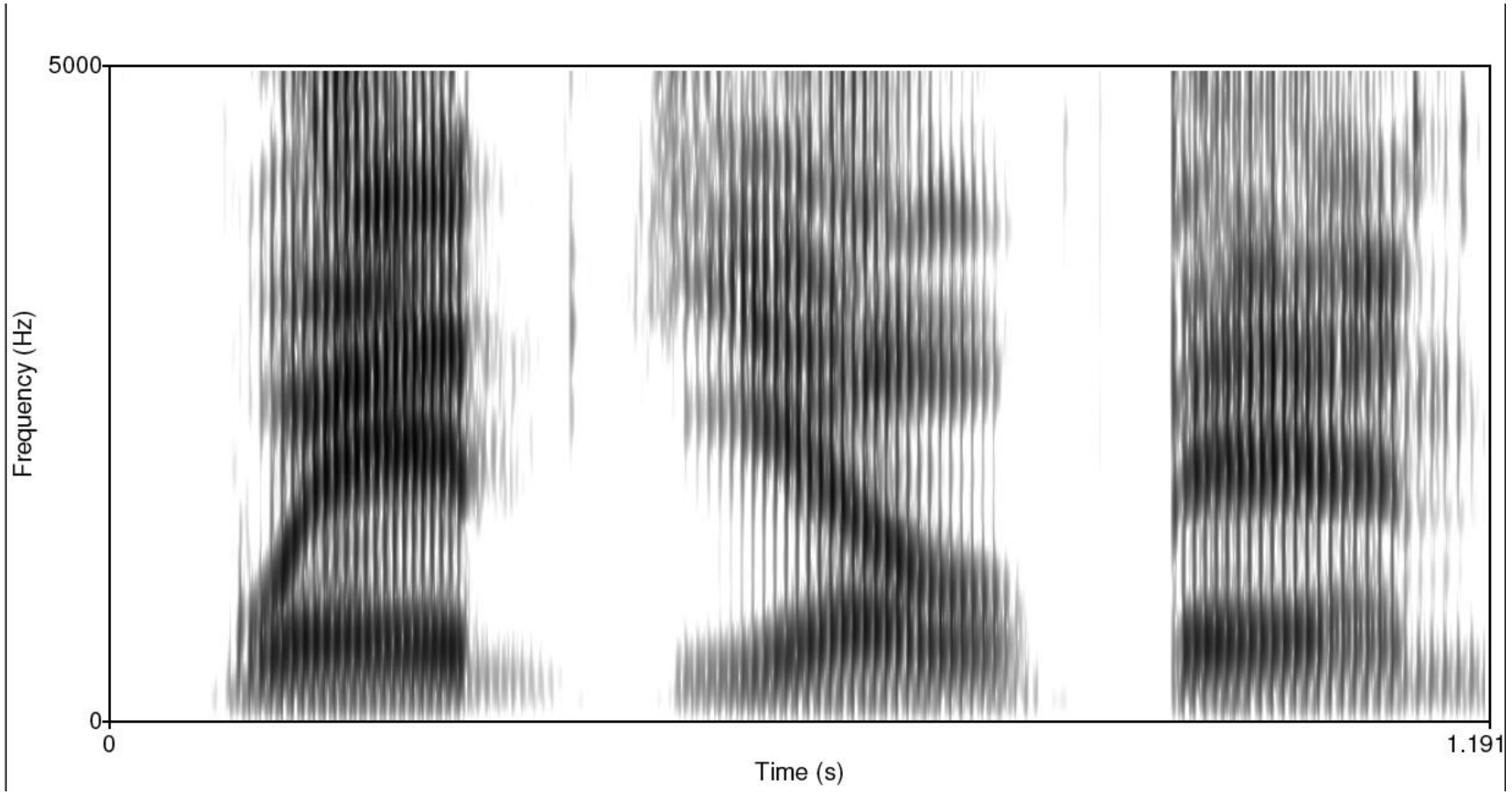
- Для много классовой классификации выходы сети представляем вероятностями $p_j = \frac{\exp(x_j)}{\sum_k \exp(x_k)}$
- В качестве лоса используем кросс энтропию $L(x, y; W, b) = -\sum_{k=1}^{K} (y = k) \log f\ (x)_k$

Обучение DNN

- Две фазы обучения
  - Прямой проход
  - Обратный проход
- Прямой проход – подаем вход (фичи).
  - Рассчитываем активации на слоях и выходе сети
  - Рассчитываем полученную ошибку
- Обратный проход:
  - Считаем градиент ошибки по входу $\partial L/\partial x$
  - Используем цепное правило для распространения градиента по сети и обновления весов сети
  - $\Delta w_{ij}\ (t) = \alpha \Delta w_{ij}(t\ -\ 1) - \epsilon \frac{\partial C}{\partial w_{ij}(t)}$.

Неоднородность звучания фонем в разных словах

w **eh** d y **eh** l b **eh** n

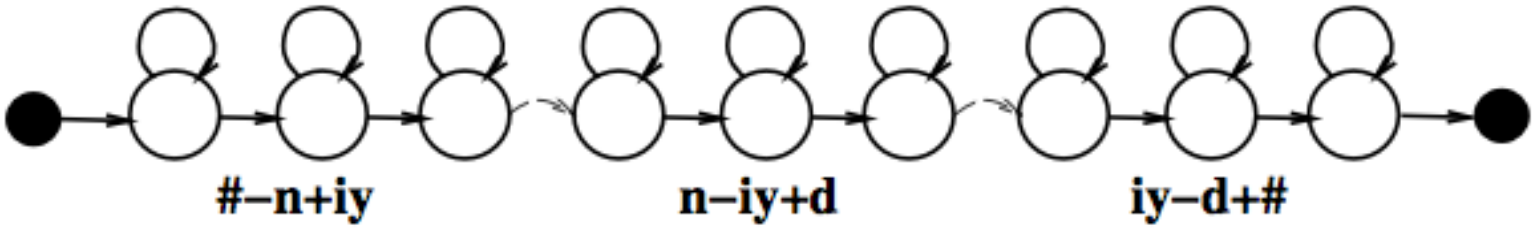


Моделирование фонетического контекста

- Самый сильный фактор, влияющий на фонетическую изменчивость это соседние фонемы
- Как моделировать это в HMM?
- Идея: иметь модели фонем, которые являются специфическими для контекста.
- Вместо контекстно-независимых (CI) фонем
- Мы будем использовать контекстно-зависимые (CD) фонемы

CD фонемы – трифоны (triphones)

- Каждый трифон инкапсулирует информацию о предыдущих и следующих фонемах
- Монофоны:
  - p, t, k
- Трифоны
  - iy-p+aa
  - a-b+c - значит “фонема a предшествует b за которой стоит c”



Нужно учитывать трифоны в модели

Моделирование границ слов

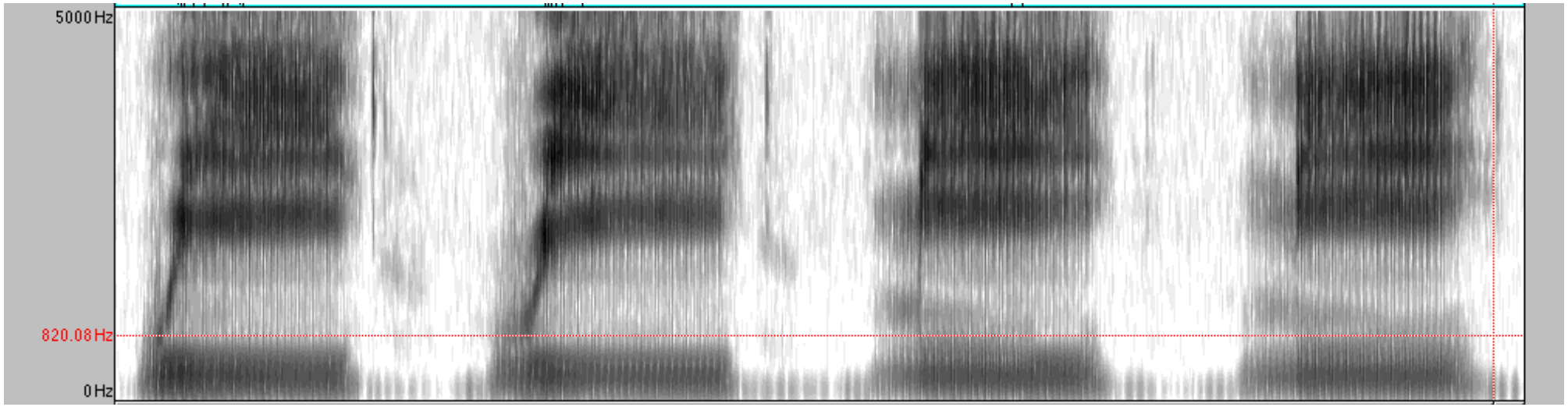
- Внутри словные контекстно зависимые (CD) модели
  ‘OUR LIST’:
  SIL AA+R AA-R L+IH L-IH+S IH-S+T S-T
- Между словные контекстно зависимые (CD) модели
  ‘OUR LIST’:
  SIL-AA+R AA-R+L R-L+IH L-IH+S IH-S+T S-T+SIL
- Учет между словных трифонов делает декодирование сложнее!

Последствия кросс-словных трифонов

- Если фонетический набор - 50 фонем, то возможное количество трифонов - $50^3 = 125000$.
- В реальной жизни не весь набор трифонов используется.
- В задаче WSJ содержится примерно 20K слов
- Всего трифонов в обучающем сете 18500
- Нужна обобщающая модель!

Моделирование фонетического контекста

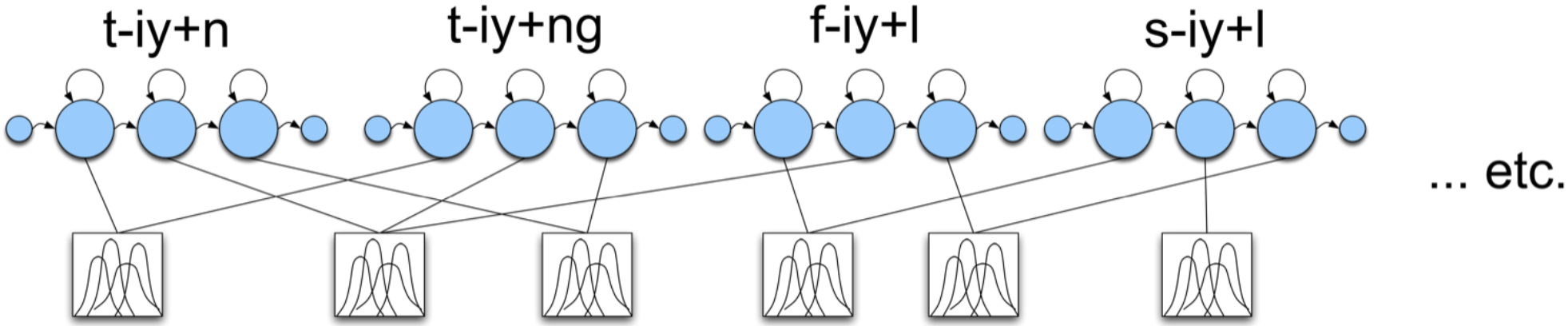
w iy r iy m iy n iy



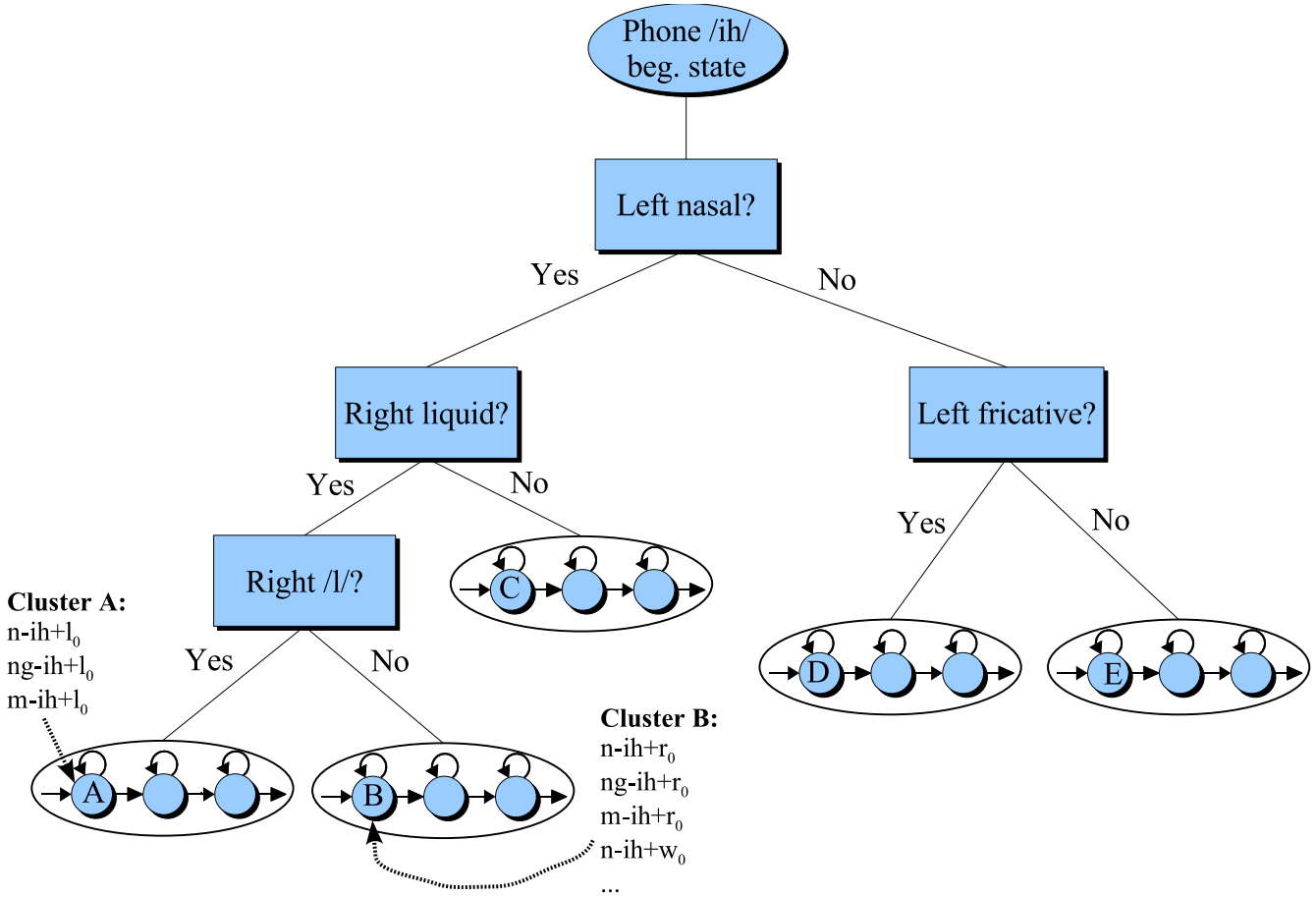
Некоторые фонемы в контексте выглядят одинаковыми

Связывание состояний *(Young et al)*

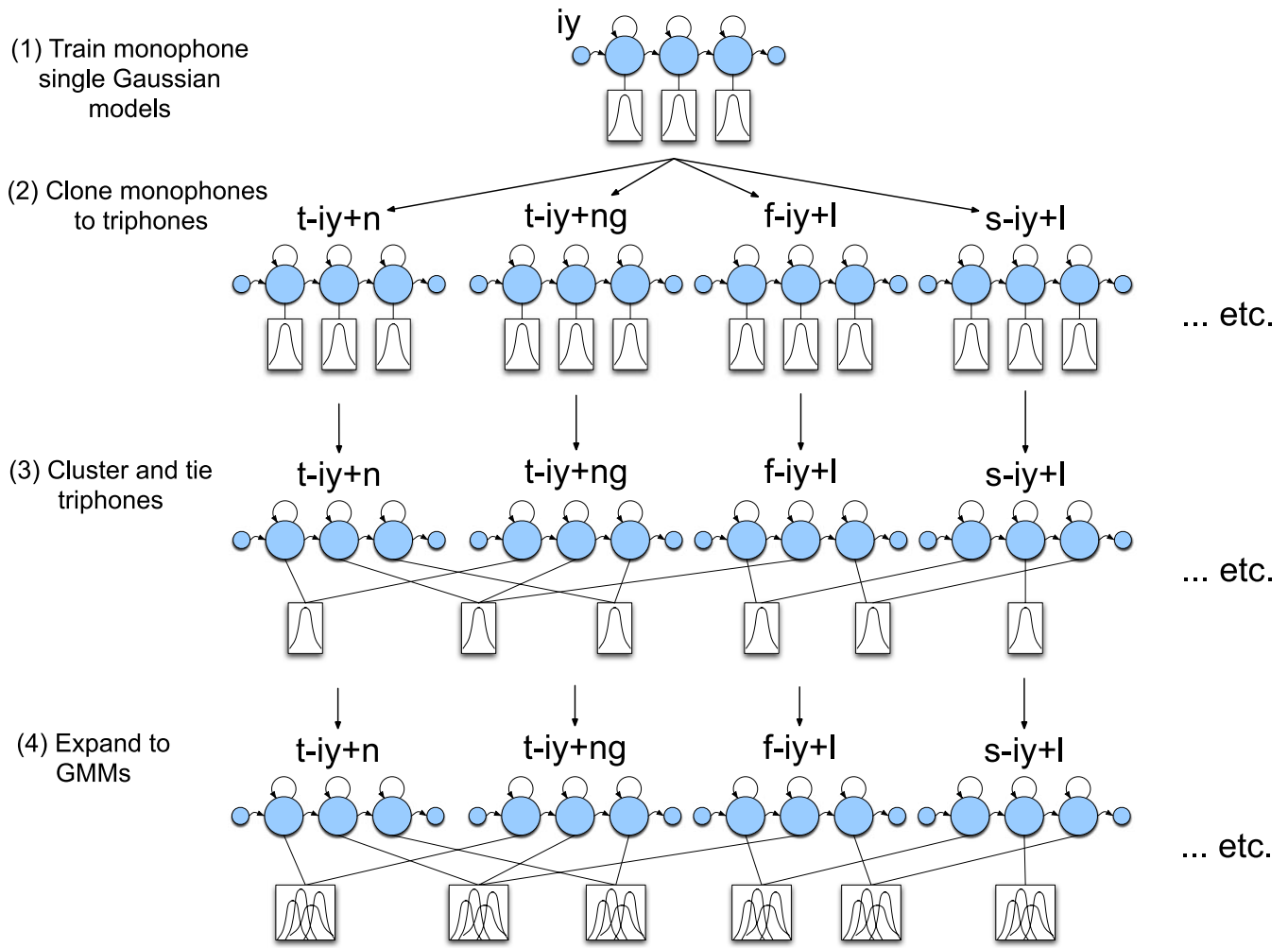
- Кластеризация трифонов на основе деревьев принятия решений
- Состояния, которые группируются вместе, будут разделять свои гауссианы
- Мы называем это «связывание состояний», поскольку эти состояния «связаны» с одним и тем же гауссовым распределением.



Дерево решений для связывания состояний



Связывание состояний *(State Tying: Young, Odell, Woodland 1994)*



Нейросеть вместо GMM

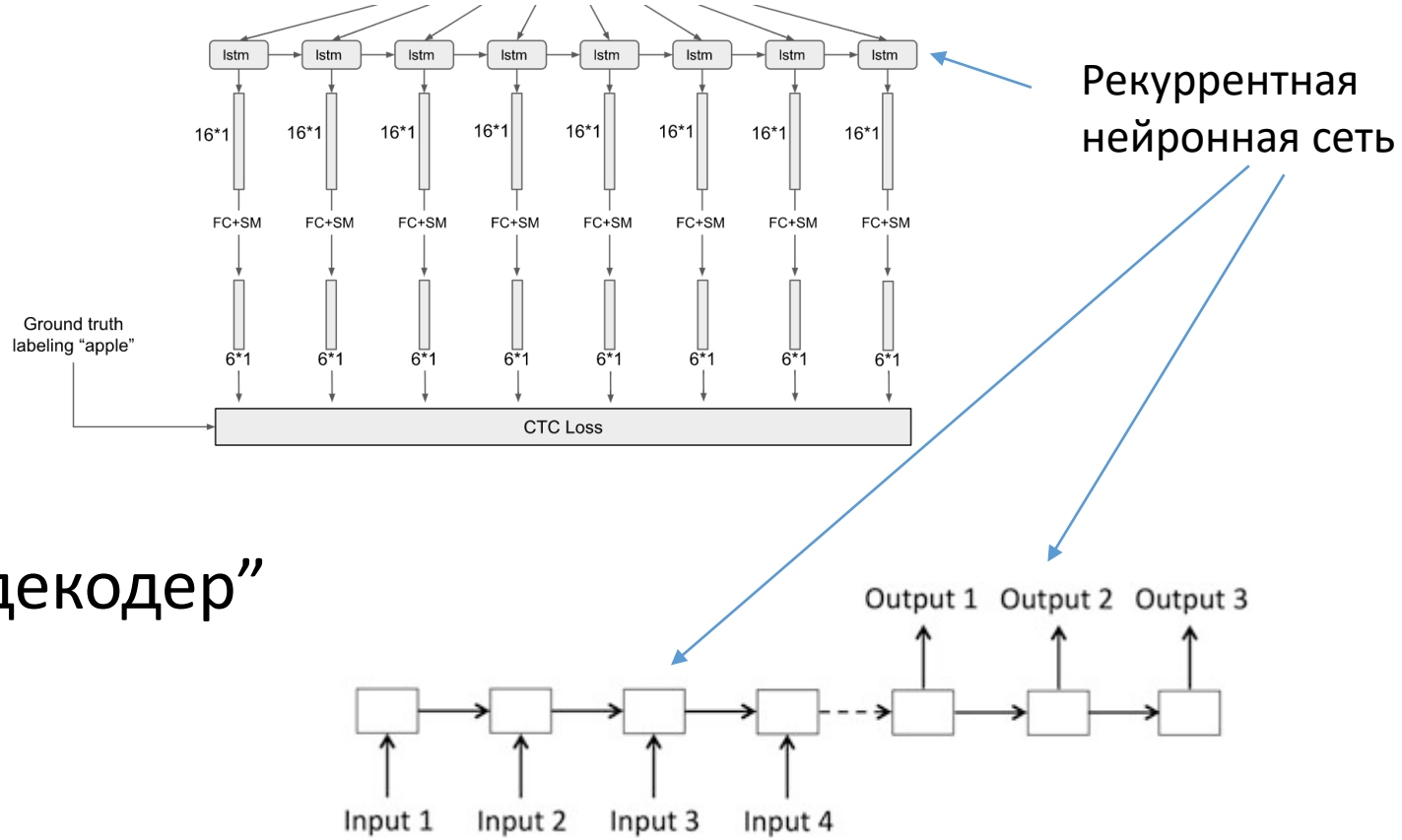
- Нейросеть лучший классификатор чем GMM, но нейросетью нельзя сделать выравнивание
- Используем нейросеть для классификации состояний и трифонов
  - Обучаем модель HMM + GMM + CD
  - На маркерах полученных из HMM+GMM +CD обучаем DNN
  - Еще раз делаем выравнивание при помощи HMM+DNN
    - F. Seide, G. Li, and D. Yu, “Conversational speech transcription using context-dependent deep neural networks.”
  - Перетренируем DNN

Основные недостатки HMM + DNN

- Множество фаз тренировки.
- Различные типы ресурсов. Например, нужны фонемные словари.
- Большое количество гиперпараметров
- Различные функции оптимизации

End-To-End архитектуры для распознавания

- Connectionist Temporal Classification (CTC)



- Парадигма “Энкодер-декодер”

Рекуррентная формула

Сеть подает на вход в момент времени *t* выход *t-1*

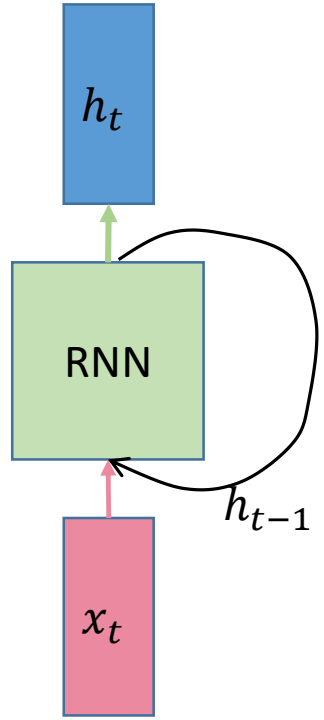
$$h_t = f_W(h_{t-1}, x_t)$$

Новое состояние

Старое состояние

Вход на шаге t

Функция с параметрами W



Рекуррентная формула

Сеть подает на вход в момент времени *t* выход *t-1*

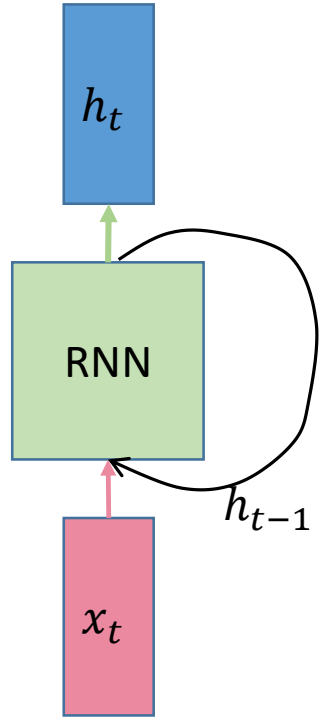
$$h_t = f_W(h_{t-1}, x_t)$$

Новое состояние

Старое состояние

Вход на шаге t

Функция с параметрами W



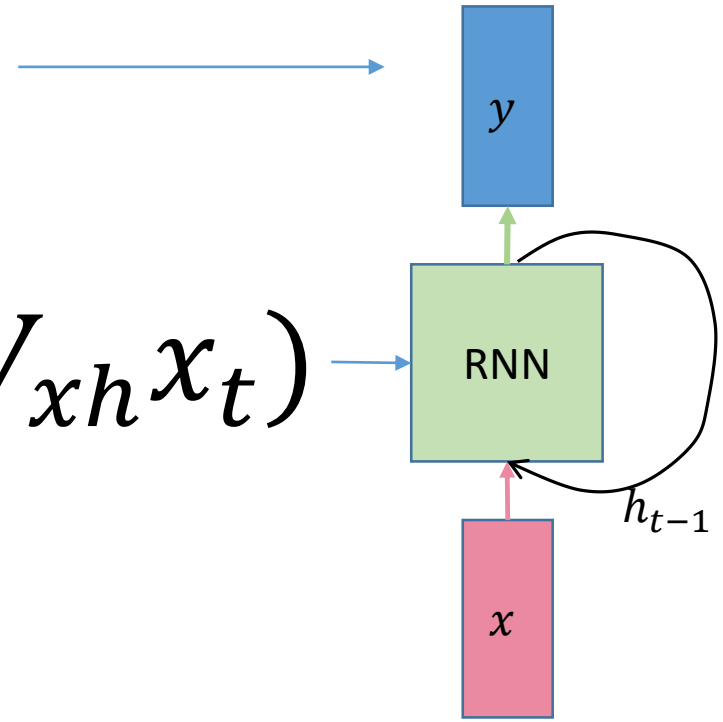
Для каждого момента времени используется одна функция и одна матрица весов

Базовая рекуррентная сеть

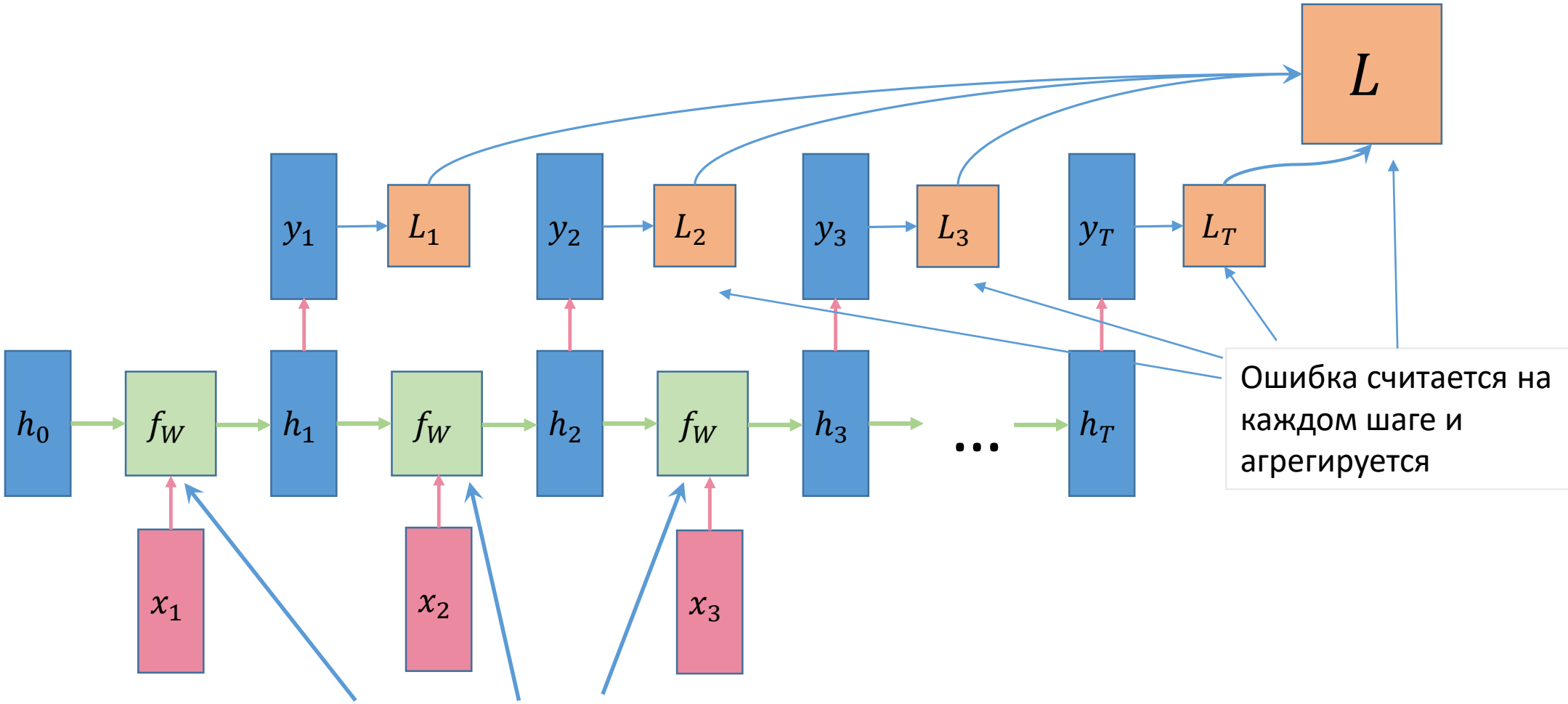
$$y_t = W_{hy}\, h_t$$

$$h_t = \tanh(W_{hh} h_{t-1} + W_{xh} x_t)$$

$$h_t = f_W\,(h_{t-1}, x_t)$$

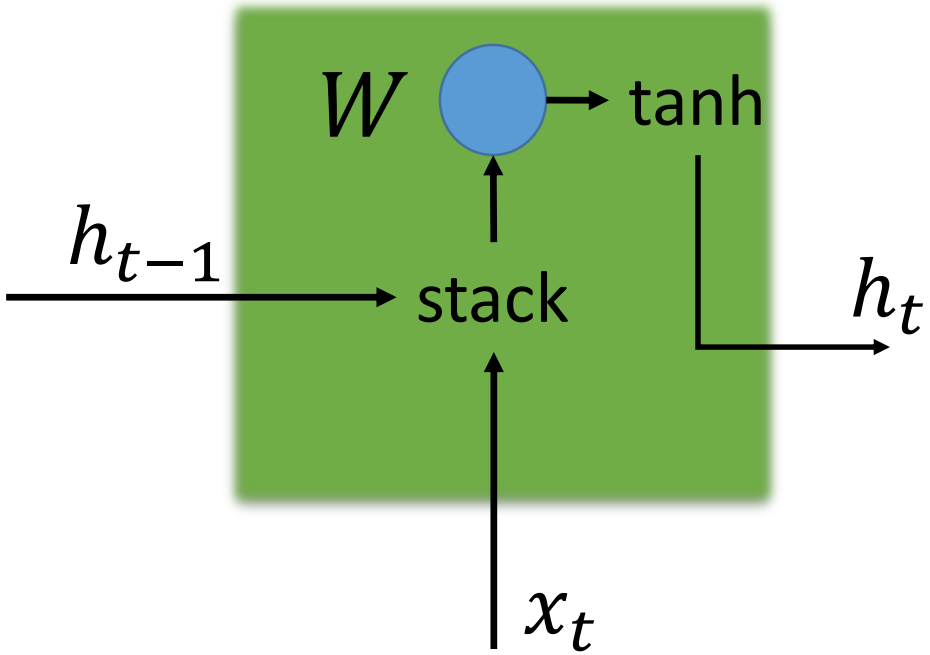


Вычислительный граф. Один ко многим



Одна матрица W на каждом шаге

Градиент через RNN



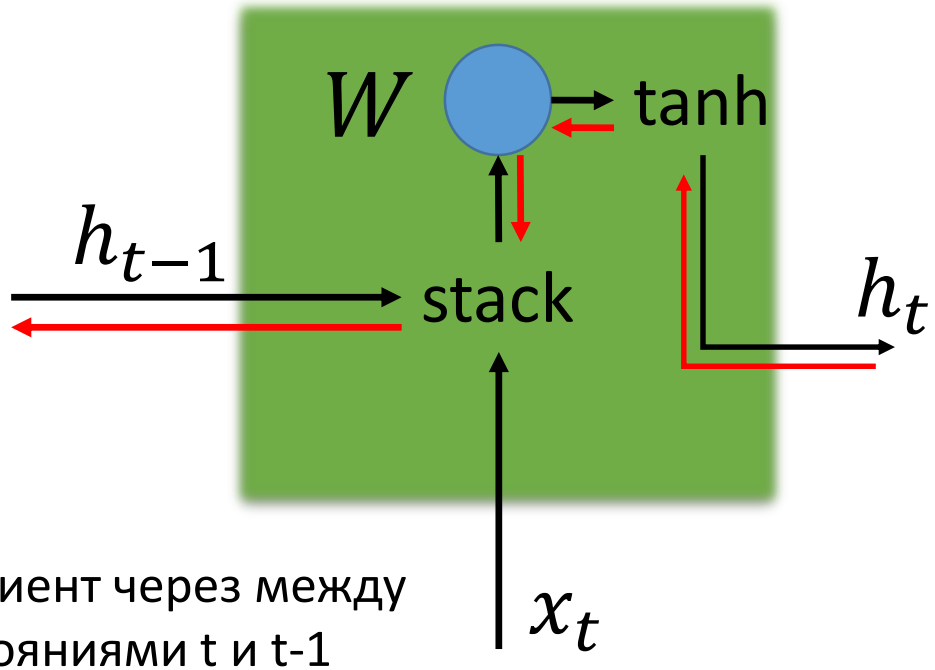
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$= \tanh((W_{hh} \quad W_{xh}) \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix})$$

$$= \tanh(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix})$$

Bengio et al, “Learning long-term dependencies with gradient descent is difficult”, IEEE Transactions on Neural Networks, 1994
Pascanu et al, “On the difficulty of training recurrent neural networks”, ICML 2013

Градиент через RNN



$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

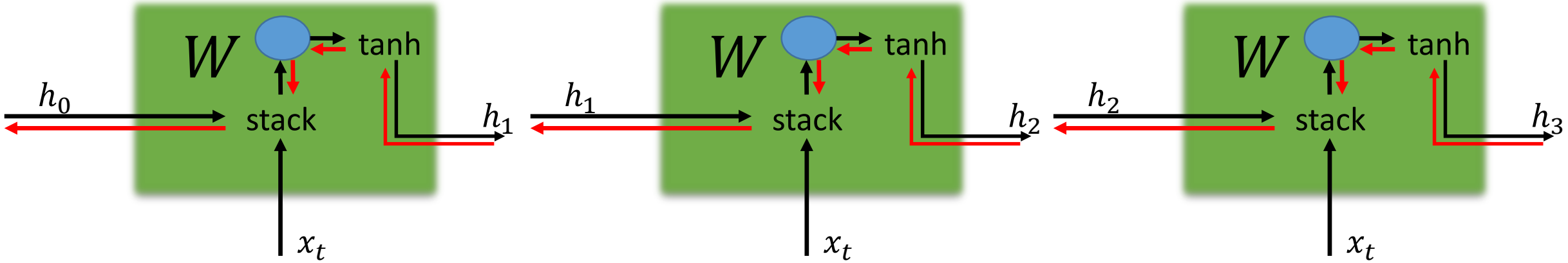
$$= \tanh((W_{hh} \quad W_{xh}) \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix})$$

$$= \tanh(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix})$$

Градиент через между состояниями t и t-1 умножается на матрицу весов W^T

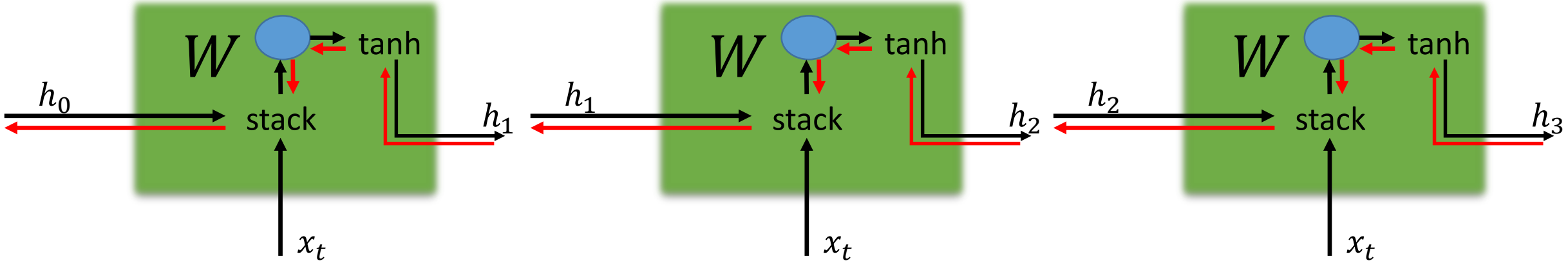
Bengio et al, “Learning long-term dependencies with gradient descent is difficult”, IEEE Transactions on Neural Networks, 1994
Pascanu et al, “On the difficulty of training recurrent neural networks”, ICML 2013

Градиент через RNN



Вычисление градиента для состояния 0 будет приводить к многократному умножению на матрицу весов *W*

Градиент через RNN

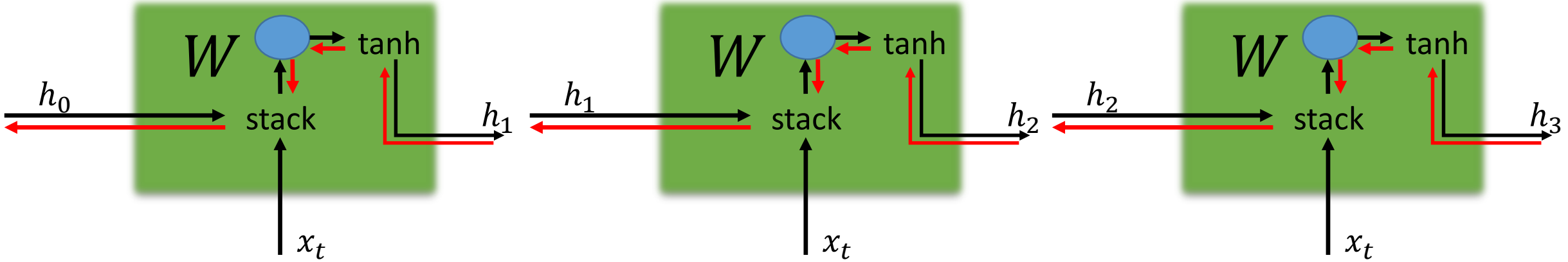


Вычисление градиента для состояния 0 будет приводить к многократному умножению на матрицу весов *W*

Значения матрицы больше 1:
Exploding gradient

Значения матрицы меньше 1:
Vanishing gradient

Градиент через RNN



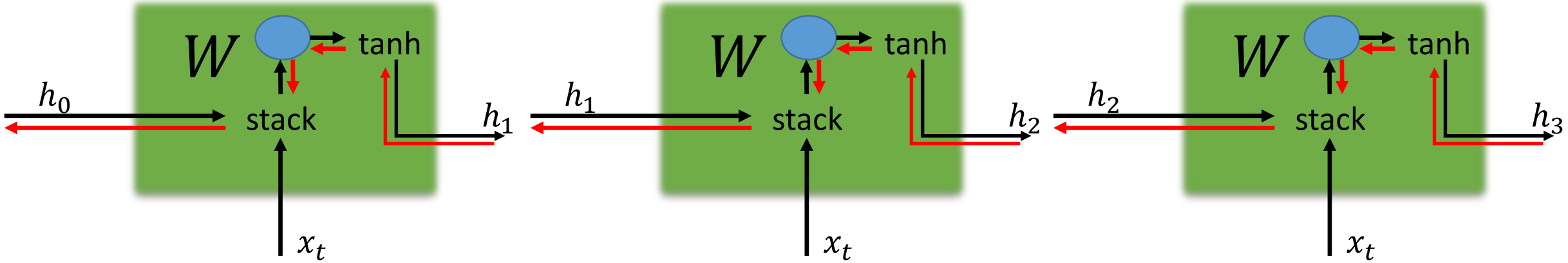
Вычисление градиента для состояния 0 будет приводить к многократному умножению на матрицу весов *W*

Значения матрицы больше 1:

Exploding gradient

Gradient clipping

Градиент через RNN



Вычисление градиента для состояния 0 будет приводить к многократному умножению на матрицу весов *W*

Значения матрицы меньше 1:

Vanishing gradient

Нужно менять архитектуру RNN

Long Short Term Memory (LSTM)

Vanilla RNN

$$h_t = \tanh\left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$

LSTM

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

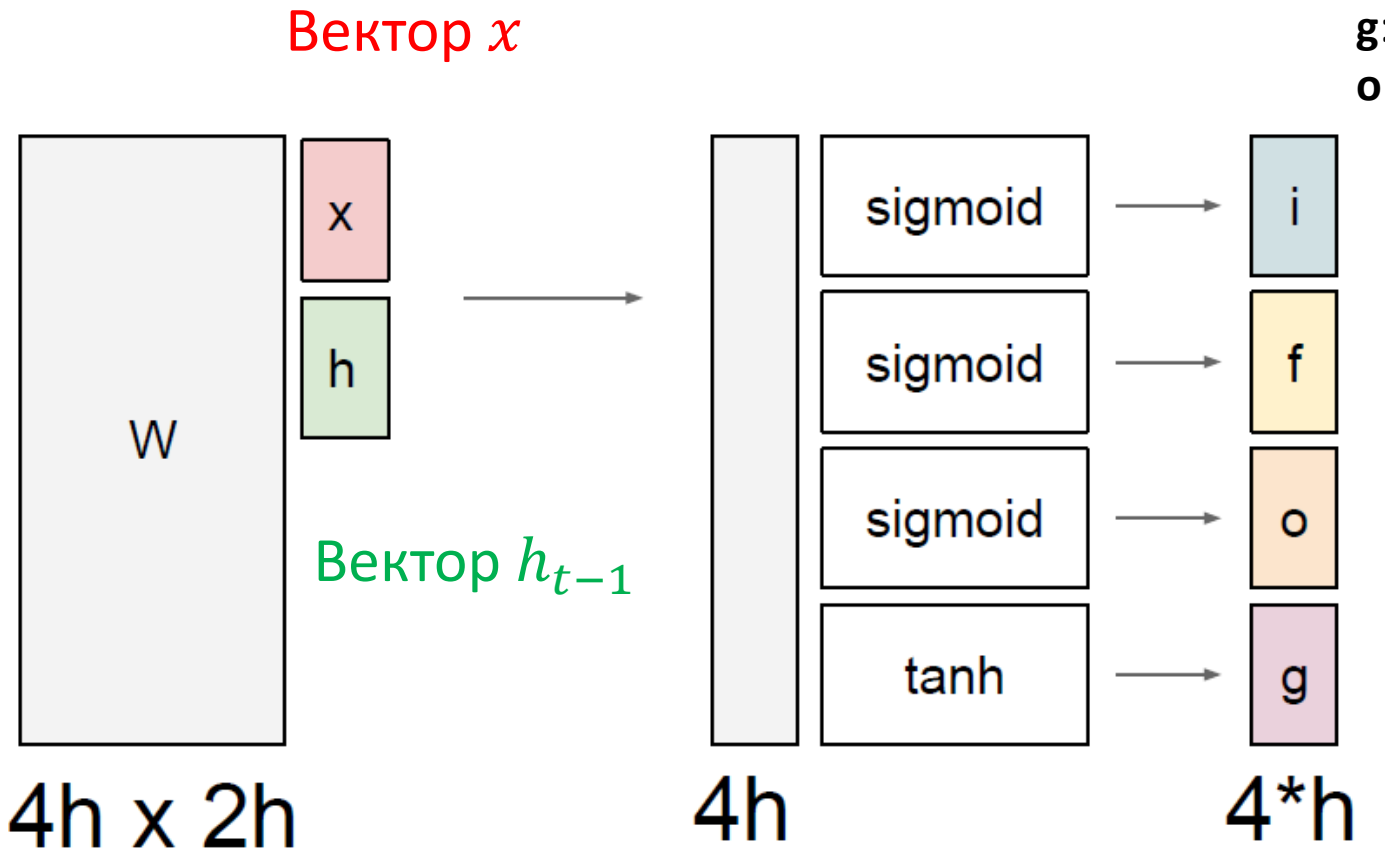
$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

Hochreiter and Schmidhuber, “Long Short Term Memory”, Neural Computation
1997

Long Short Term Memory (LSTM)

f: Forget gate, Забывать ли состояние элемента
i: Input gate, Писать ли состояние элемента
g: Gate gate, Сколько писать в элемент
o: Output gate, Сколько выводить из элемента

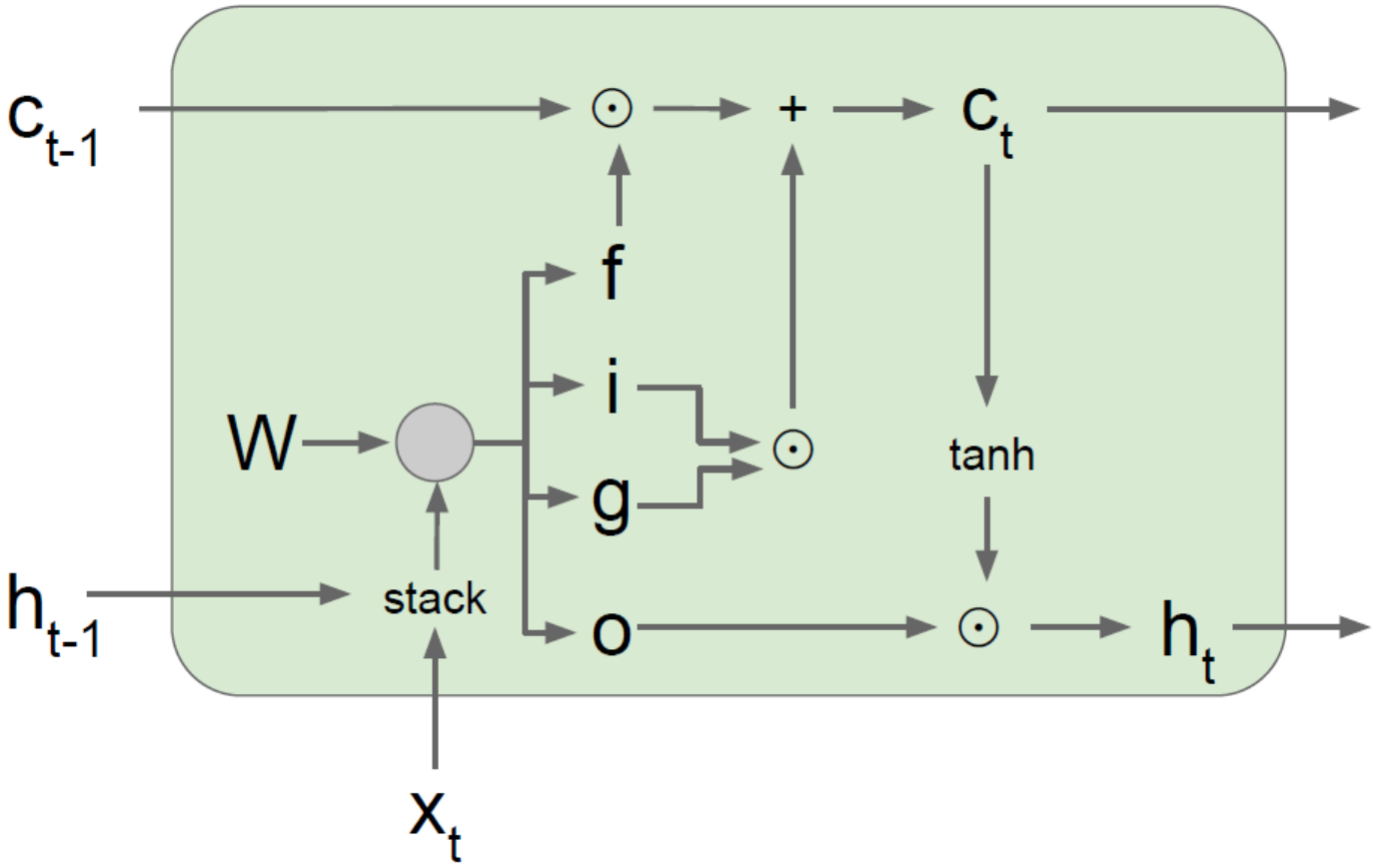


$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

Long Short Term Memory (LSTM)

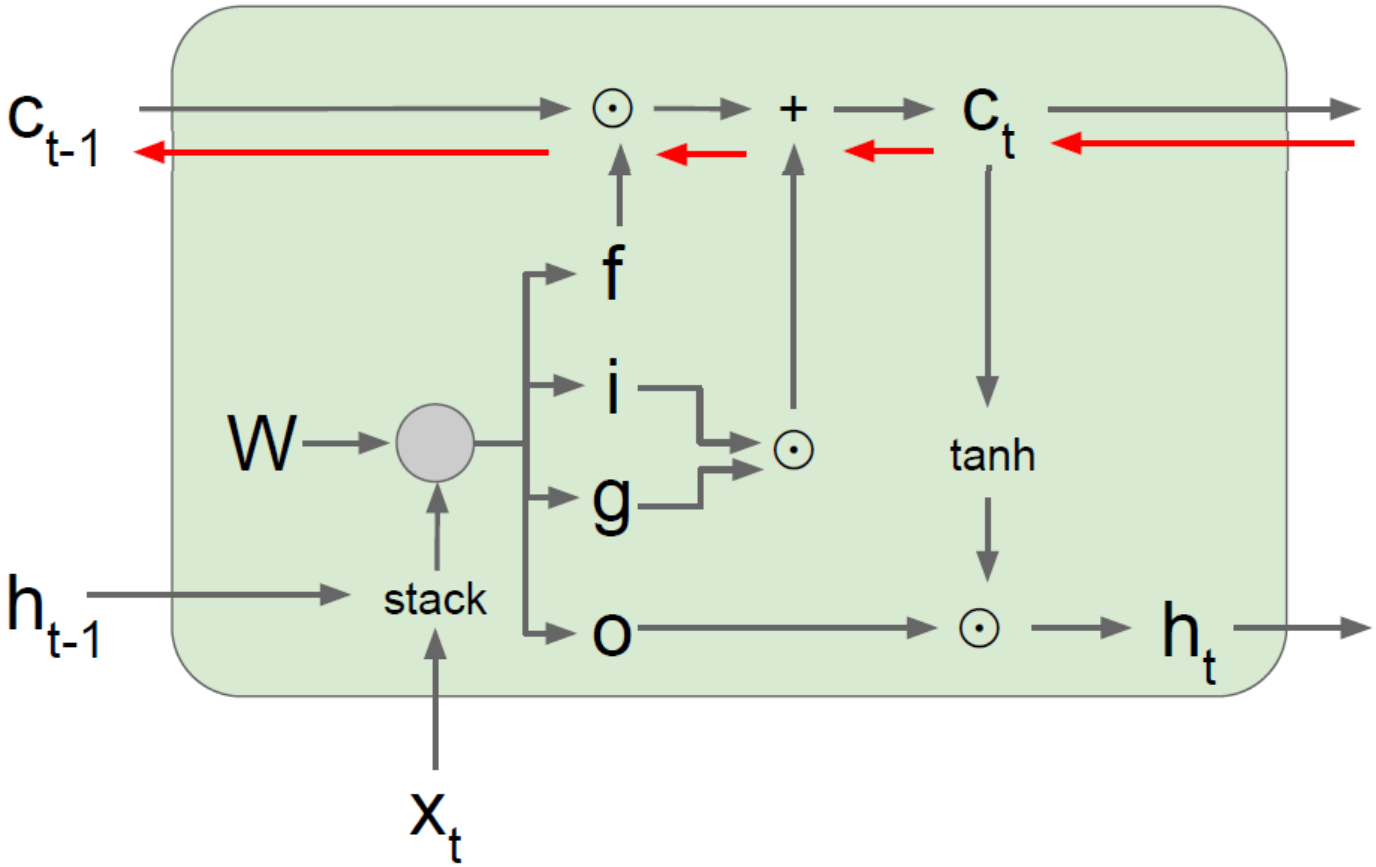


$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

LSTM. Gradient Flow



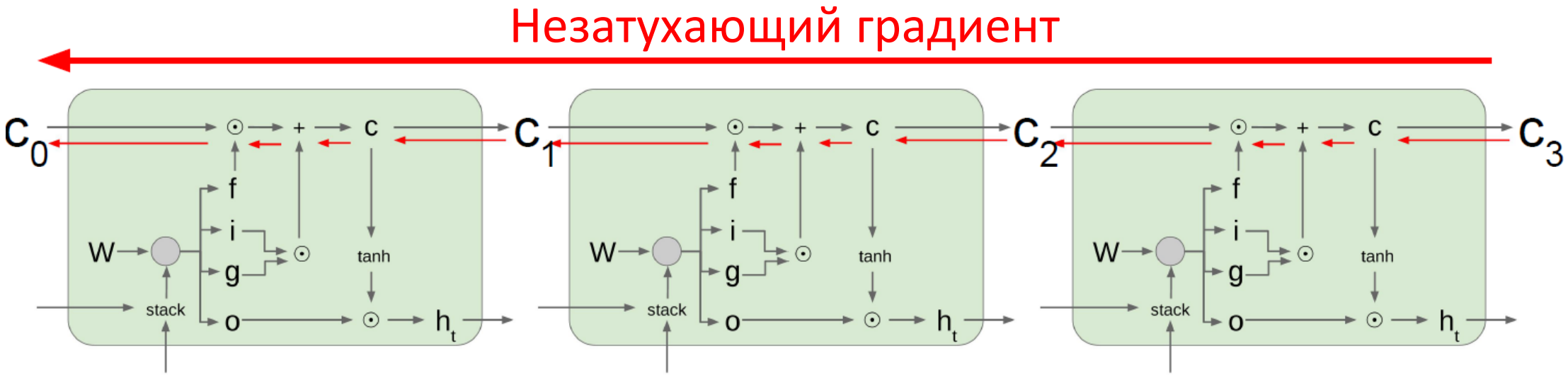
Обратный проход от c_t до c_{t-1} перемножаются только с f без матрицы W

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

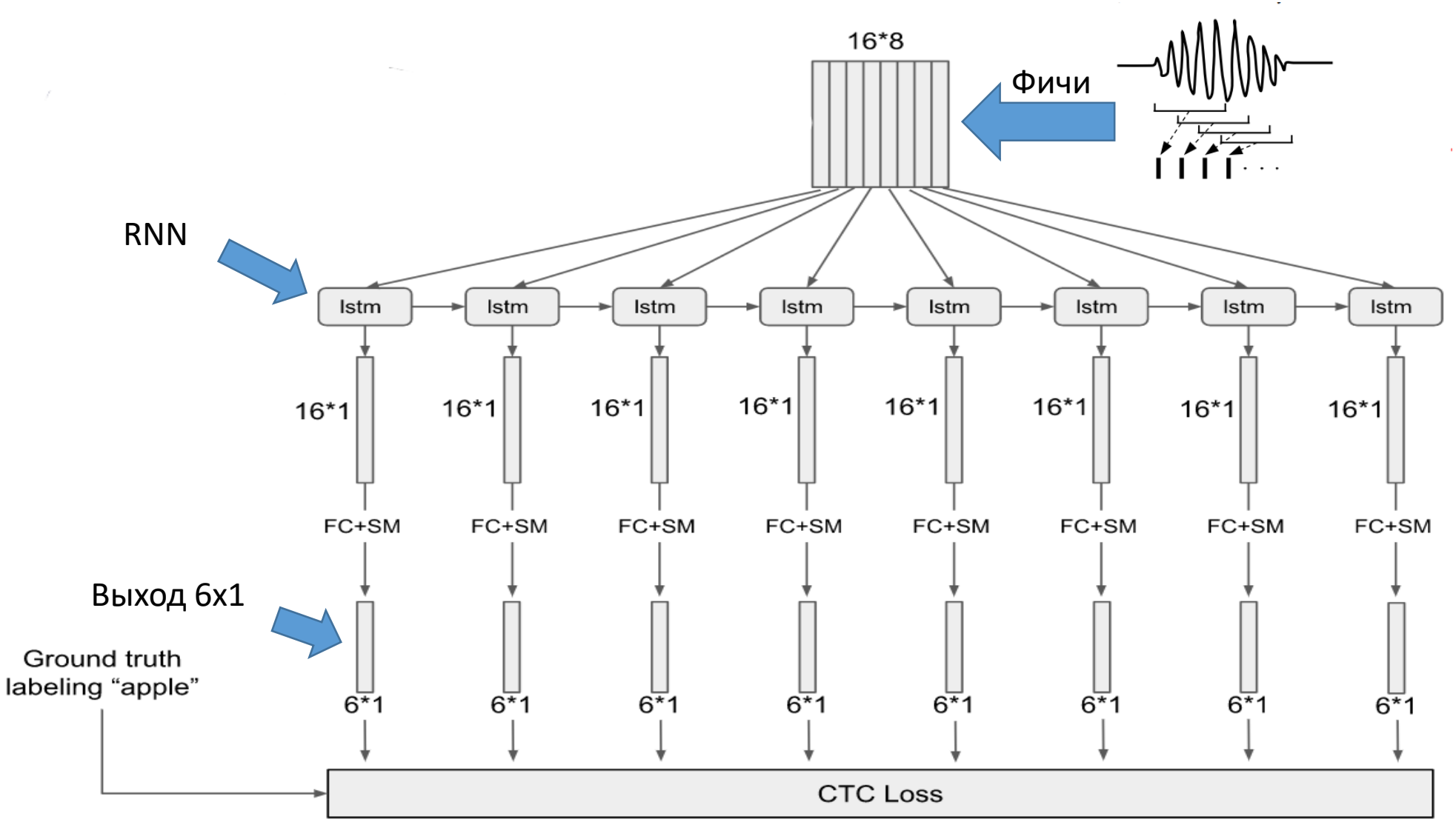
LSTM. Gradient Flow



Connectionist Temporal Classification (CTC)

- Проблема RNN
  - Длинна входной последовательности больше либо равна длине выходной последовательности
    - $\boldsymbol{x} = (x_1, x_2, \dots, x_T)$ – входная последовательность длинной T
    - $\boldsymbol{z} = (z_1, z_2, \dots, z_U)$ – выходная последовательность длинной U
    - U ≤ T
  - Целевые метки берутся из алфавита *L*
  - Нет априорного способа выровнять **x** и **z**
- Наша цель использовать тренировочное множество S, чтобы построить $h: X \rightarrow Z$

CTC loss


16*8
Фичи
RNN
lstm
lstm
lstm
lstm
lstm
lstm
lstm
lstm
16*1
16*1
16*1
16*1
16*1
16*1
16*1
16*1
FC+SM
FC+SM
FC+SM
FC+SM
FC+SM
FC+SM
FC+SM
FC+SM
Выход 6x1
Ground truth
labeling “apple”
6*1
6*1
6*1
6*1
6*1
6*1
6*1
6*1
CTC Loss

CTC loss

- CTC loss – это “softmax” слой $p_l = \frac{\exp(x_l)}{\sum_k \exp(x_k)}$
- Количество выходов слоя на 1 больше, чем всего маркеров ***L***
- Активация первых |L | элементов слоя интерпретируется как вероятность
- Активация дополнительного юнита интерпретируется как отсутствие маркера. “blank”

CTC loss

- Для входной последовательности **x** длинной **T**
  - Задаем *RNN* с *m* входами, *n* выходами и *w* – вектор весов как непрерывное отображение N_w: $(R_m)^T \rightarrow (R_n)^T$
  - Тогда $y = N_w(x)$ – последовательность выходов RNN
- y_k^t - активация выходного элемента *k* в момент времени *t*
- y_k^t - вероятность пронаблюдать маркер *k* в момент времени *t*
  - Определяет распределение по множеству ${L'}^T$ последовательностей длинны T над алфавитом $L' = L \cup \{blank\}$:
    - $p(\pi|x) = \prod_{t=1}^{T} y_{\pi_t}^t$, $\forall \pi \in {L'}^T$.
- Элементы ${L'}^T$ - это пути π

CTC loss - пути

Path1: “ap-pl-ee” —B(“ap-pl-ee”)→ Labeling: “apple”

$$p(\text{“}ap-pl-ee\text{”}) = y_a^1 \cdot y_p^2 \cdot y_-^3 \cdot y_p^4 \cdot y_l^5 \cdot y_-^6 \cdot y_e^7 \cdot y_e^8$$

CTC loss - пути

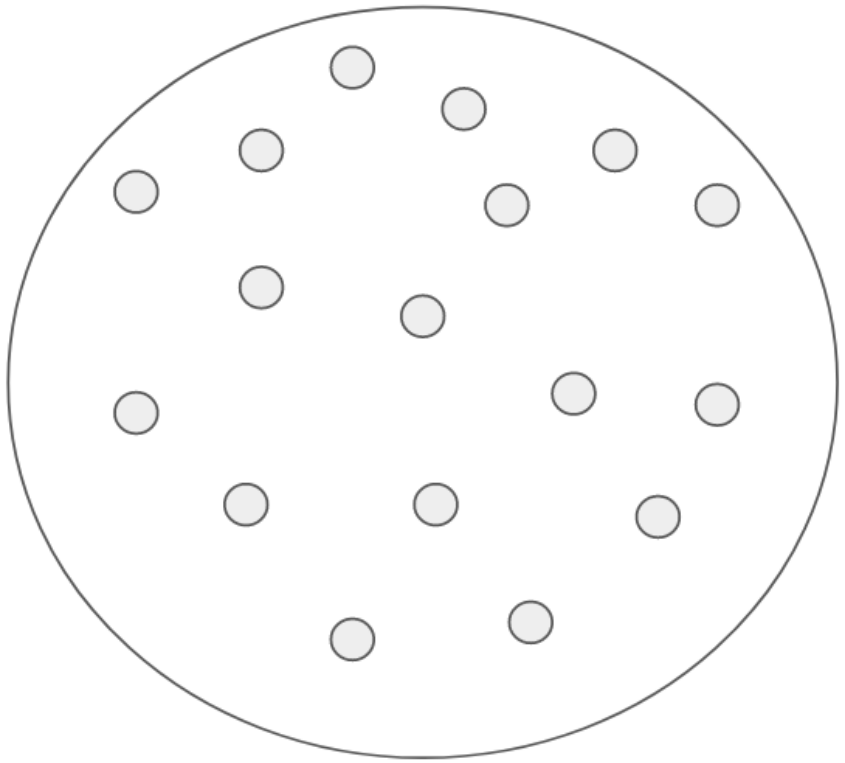
Path1: “ap-pl-ee” —B(“ap-pl-ee”)→ Labeling: “apple”

$$p(\text{“}ap-pl-ee\text{”}) = y_a^1 \cdot y_p^2 \cdot y_-^3 \cdot y_p^4 \cdot y_l^5 \cdot y_-^6 \cdot y_e^7 \cdot y_e^8$$

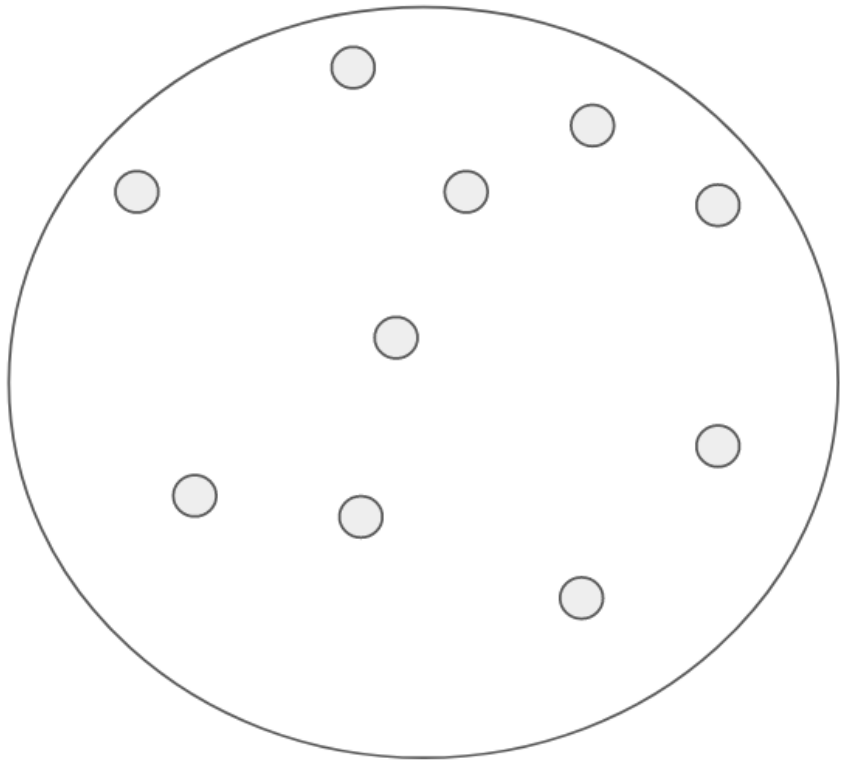
Path2: “aapp--le” —B(“aapp--le”)→ Labeling: “aple”

$$p(\text{“}aapp--le\text{”}) = y_a^1 \cdot y_a^2 \cdot y_p^3 \cdot y_p^4 \cdot y_-^5 \cdot y_-^6 \cdot y_l^7 \cdot y_e^8$$

CTC loss. Пути

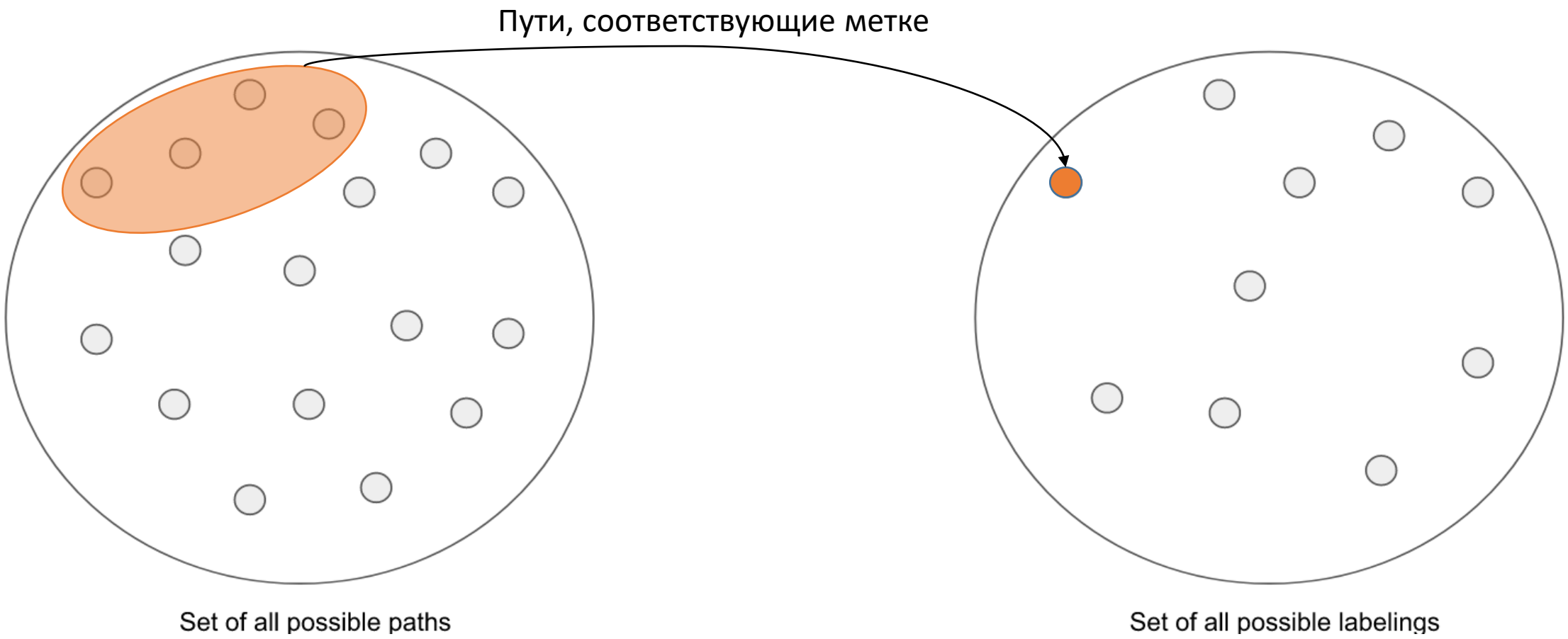


Set of all possible paths



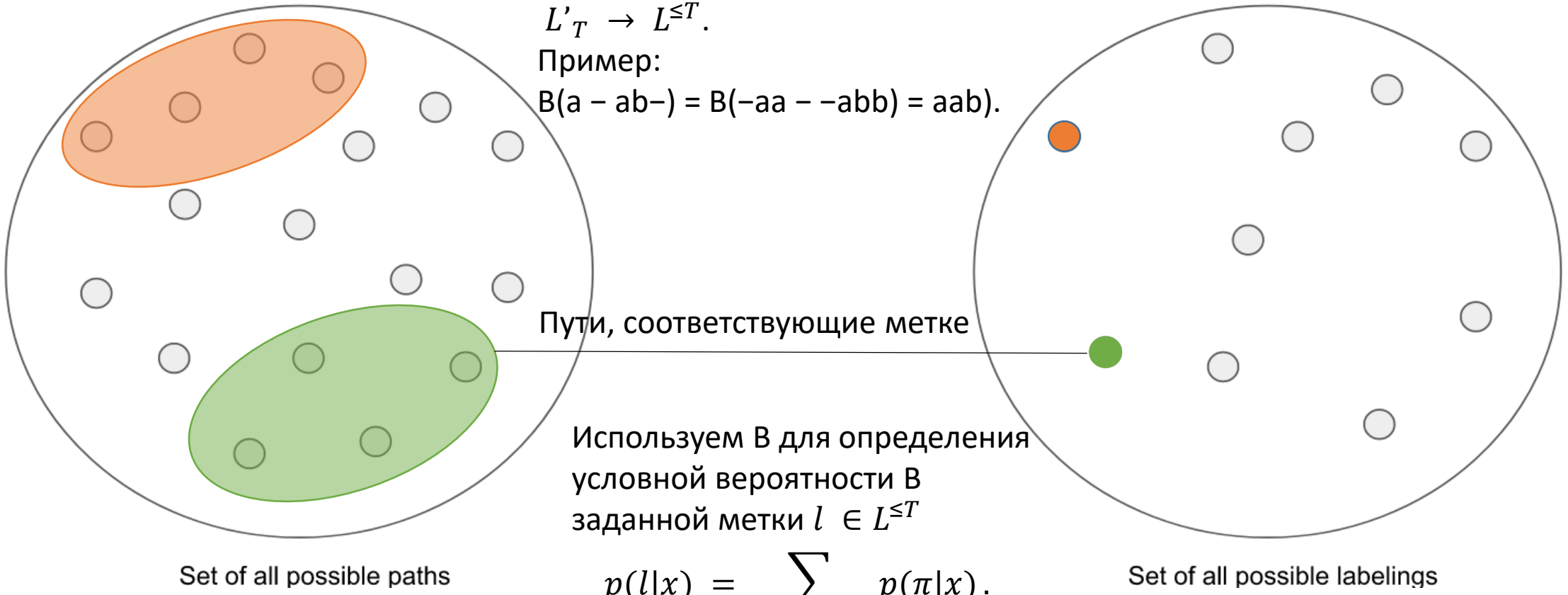
Set of all possible labelings

CTC loss. Пути



CTC loss. Пути

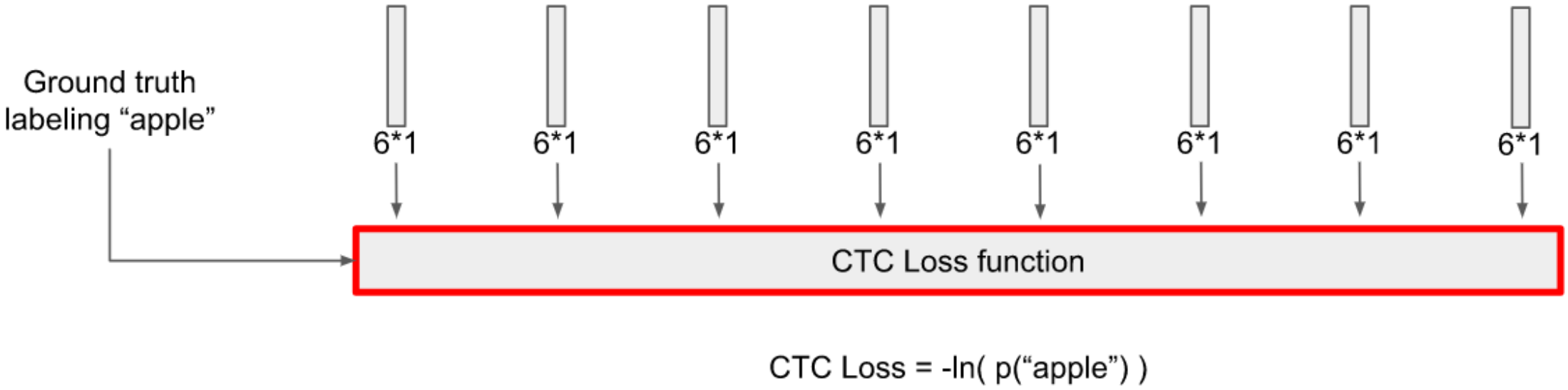
Мы хотим получить маппинг: B :
$L'_T \rightarrow L^{\leq T}$.
Пример:
B(a − ab−) = B(−aa − −abb) = aab).



Используем B для определения условной вероятности B заданной метки $l \in L^{\leq T}$

$$p(l|x) = \sum_{X_\pi \in B^{-1}(l)} p(\pi|x).$$

CTC loss



- Вероятность слова – сумма вероятностей по всем возможным путям
- 6^8 = 1 679 616 – возможных путей (случай из примера)
- Используем динамическое программирование для нахождения вероятности целевой последовательности

Нахождение возможных путей

- Аналогично прямому проходу и проходам в HMM мы рассчитываем α и β

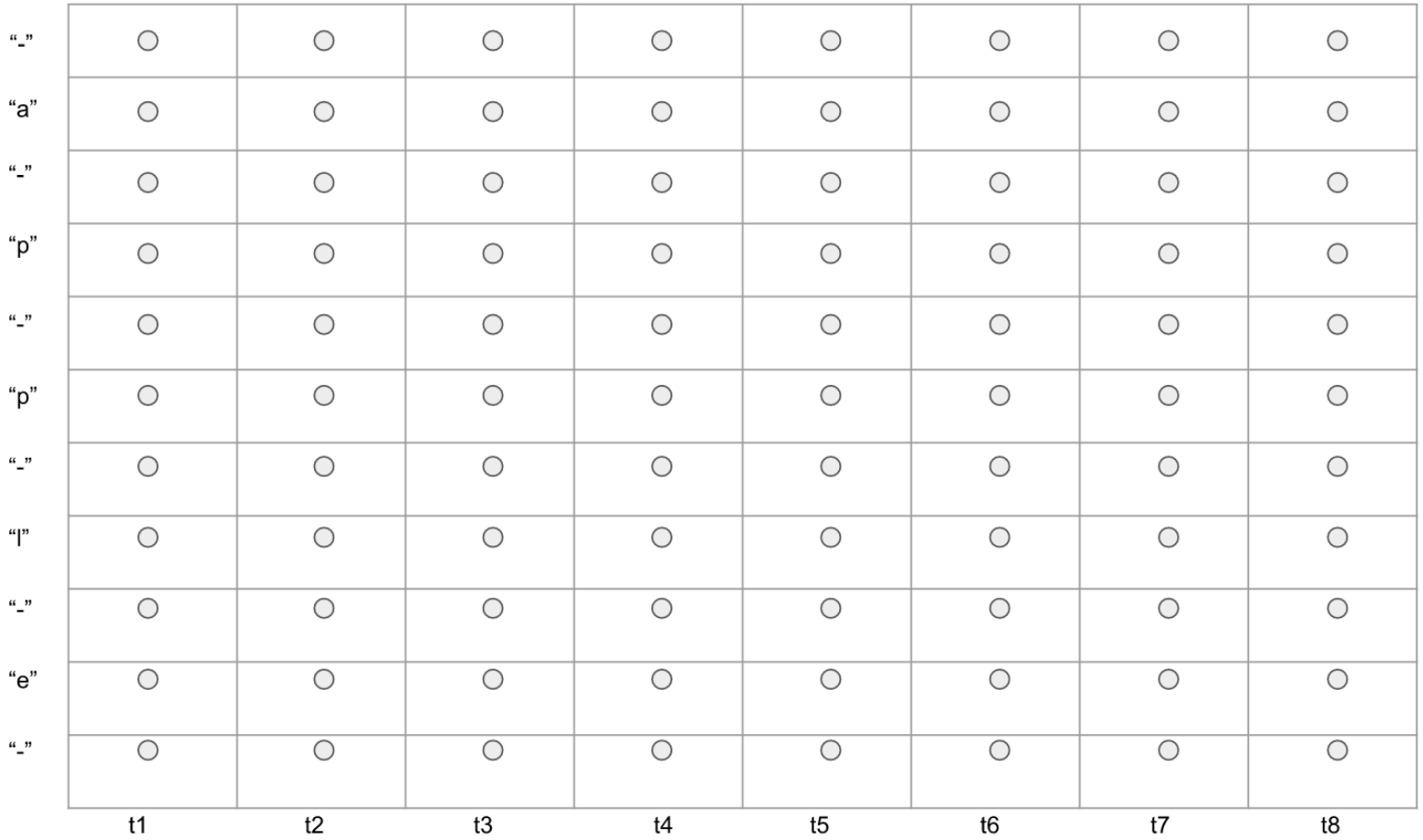
$$\alpha_t(s) \stackrel{\text{def}}{=} \sum_{\substack{\pi \in N^T: \\ \mathcal{B}(\pi_{1:t}) = \mathbf{l}_{1:s}}} \prod_{t'=1}^{t} y^{t'}_{\pi_{t'}}.$$

  - Суммарная вероятность всех путей, чей префикс заканчивается символом в позиции s в момент времени t от начала последовательности

$$\beta_t(s) \stackrel{\text{def}}{=} \sum_{\substack{\pi \in N^T: \\ \mathcal{B}(\pi_{t:T}) = \mathbf{l}_{s:|\mathbf{l}|}}} \prod_{t'=t}^{T} y^{t'}_{\pi_{t'}}$$

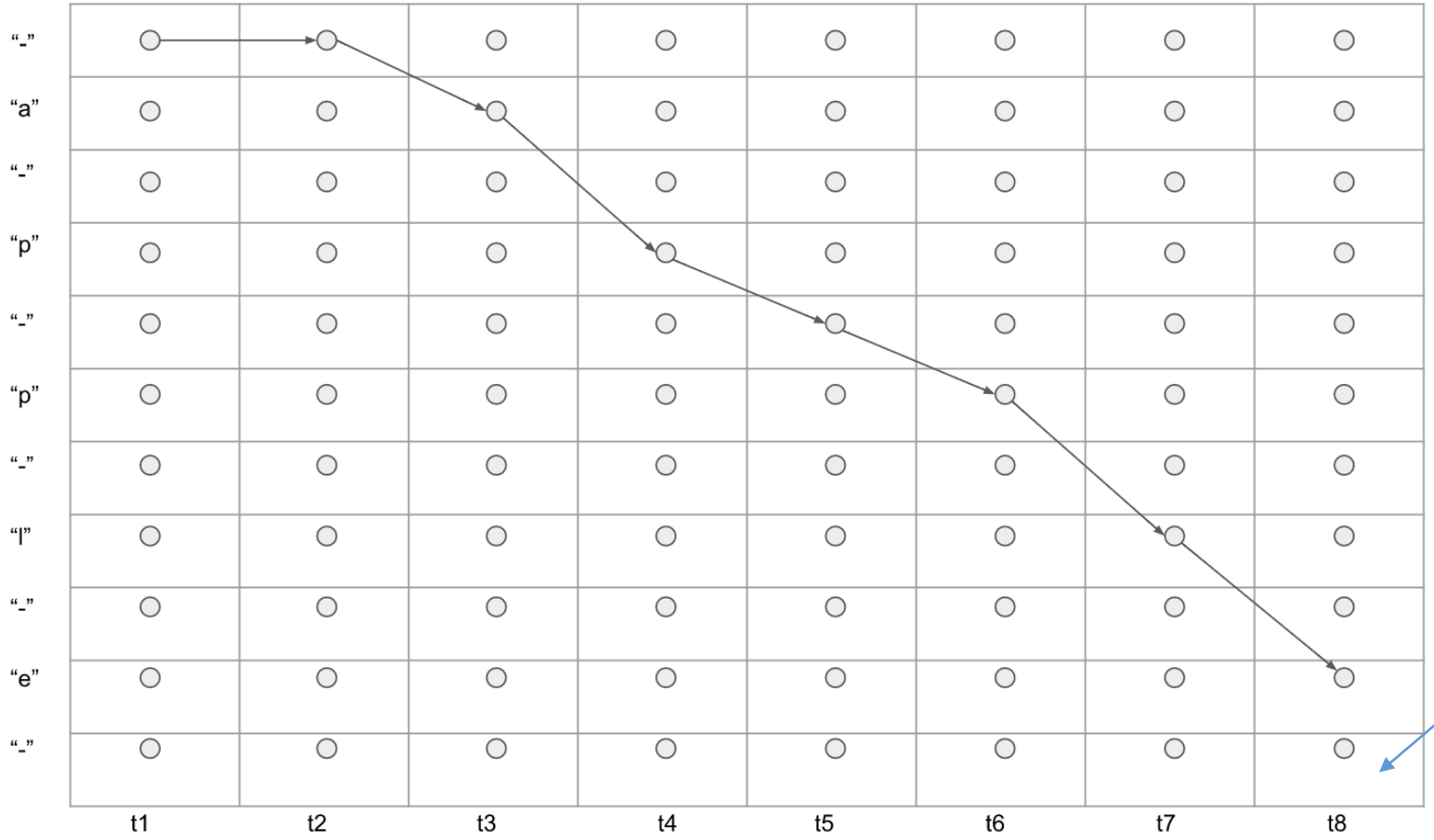
  - Суммарная вероятность всех путей, чей суффикс начинается с символом в позиции s в момент времени t

Нахождение возможных путей



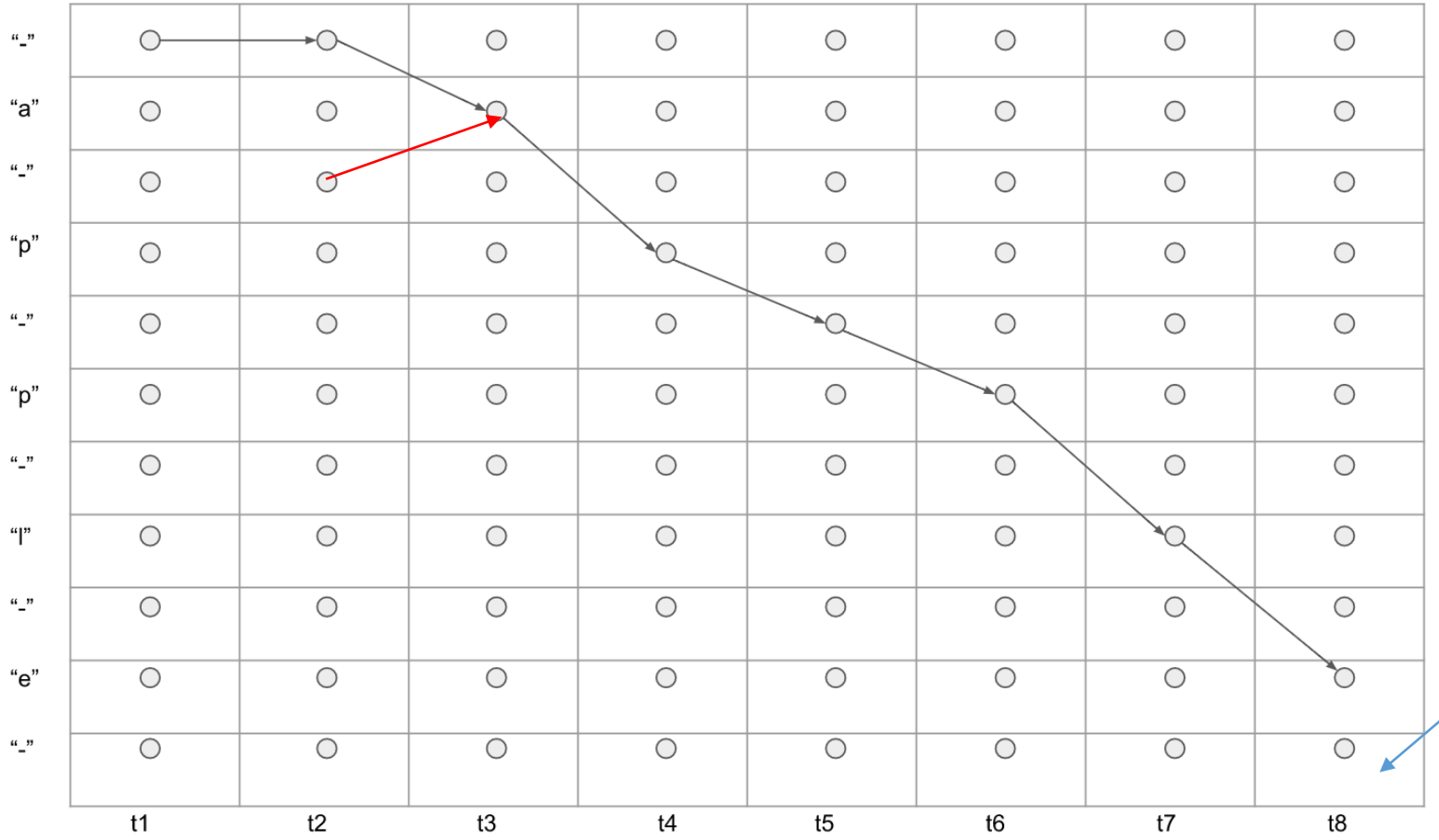
Нахождение возможных путей

Пример, путь “—ap-ple” может быть отмапирован на маркер “apple” B(--ap-ple) = “apple”



Нахождение возможных путей

Пример: невозможный переход, нельзя предсказывать предыдущий символ.



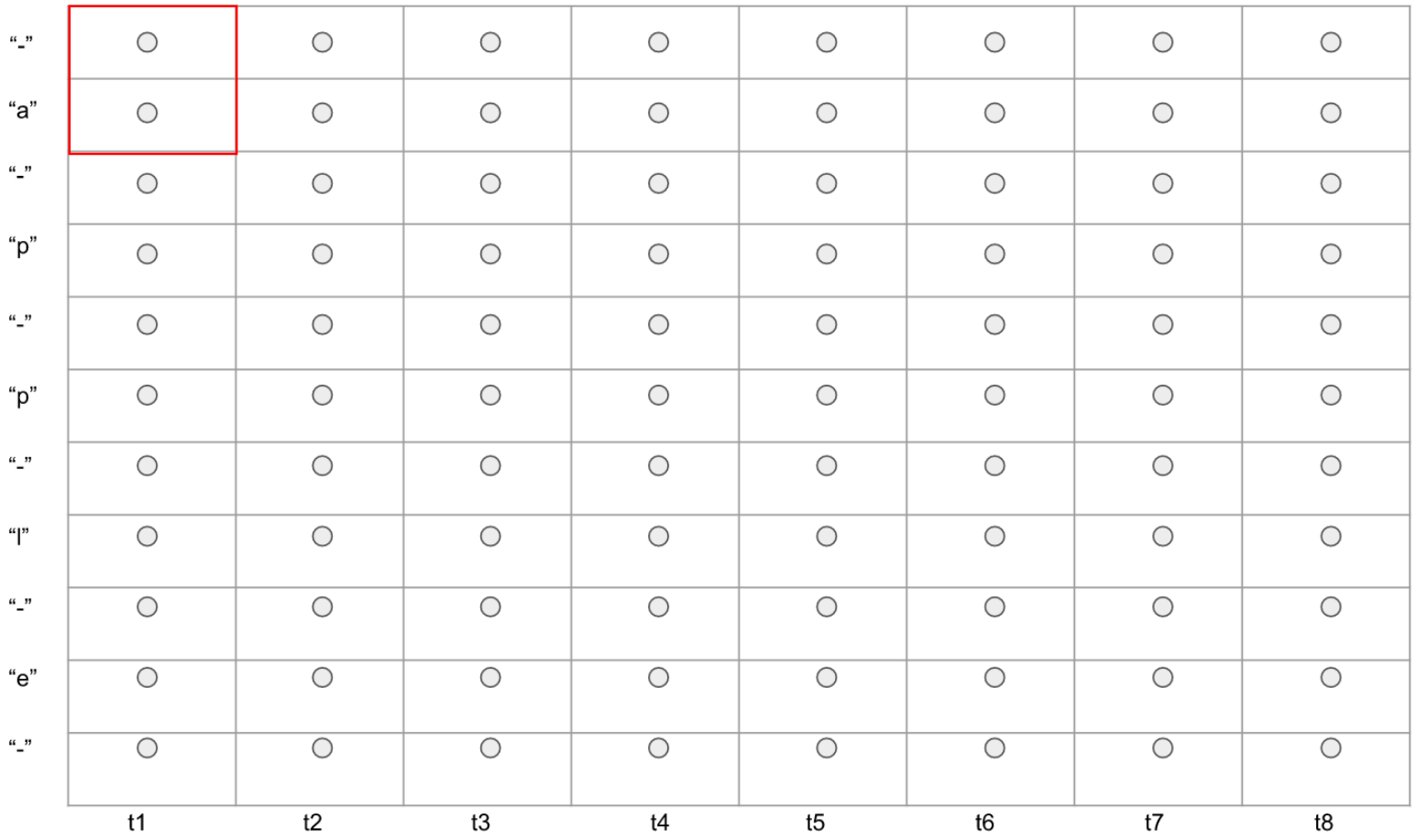
Не обязательно заканчиваем тут

Нахождение возможных путей

Начинаем либо b или с первого символа

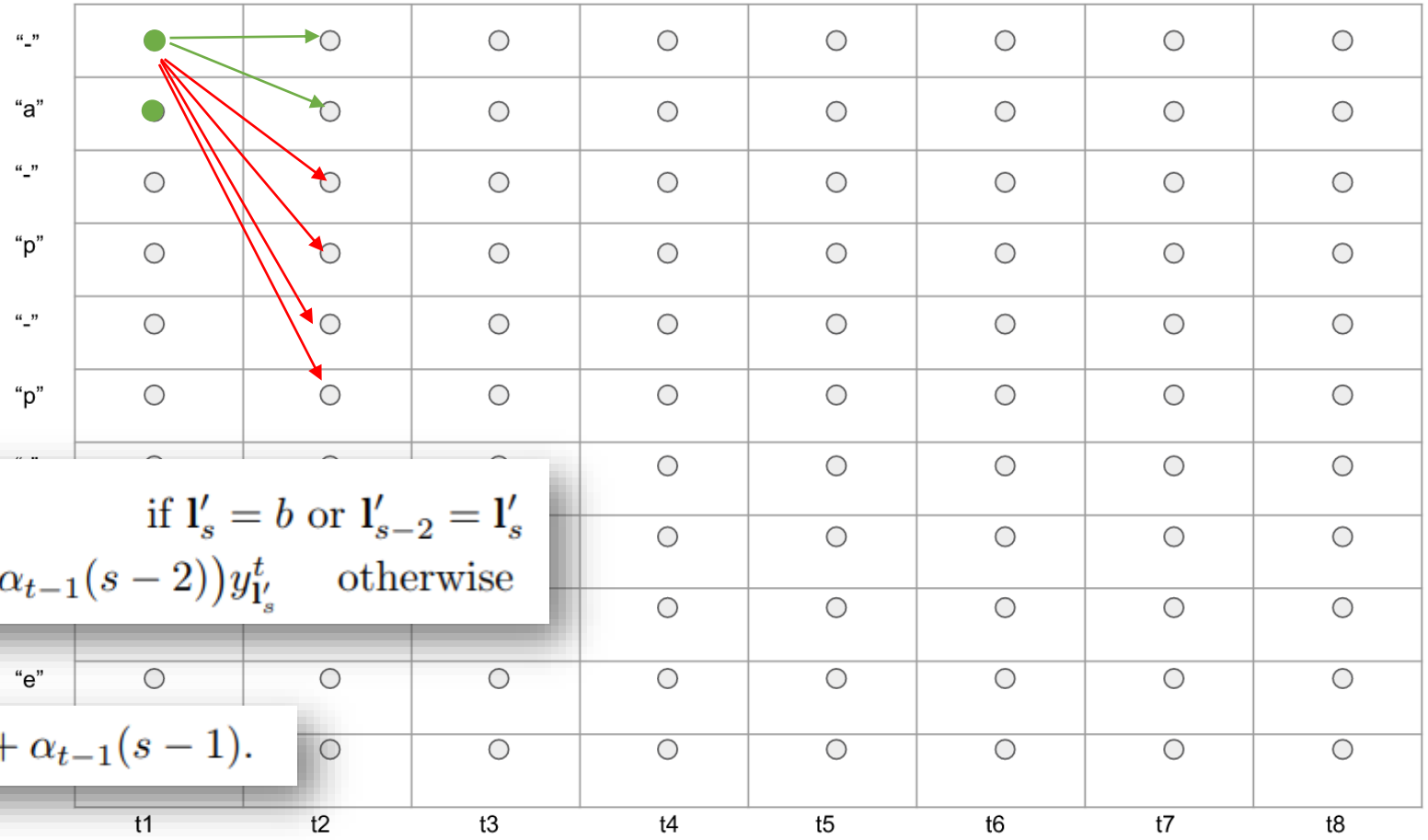
Инициализация

$$\alpha_1(1) = y_b^1$$
$$\alpha_1(2) = y_{\mathrm{I}_1}^1$$
$$\alpha_1(s) = 0, \ \forall s > 2$$



Нахождение возможных путей

Куда можно двигаться из этих точек

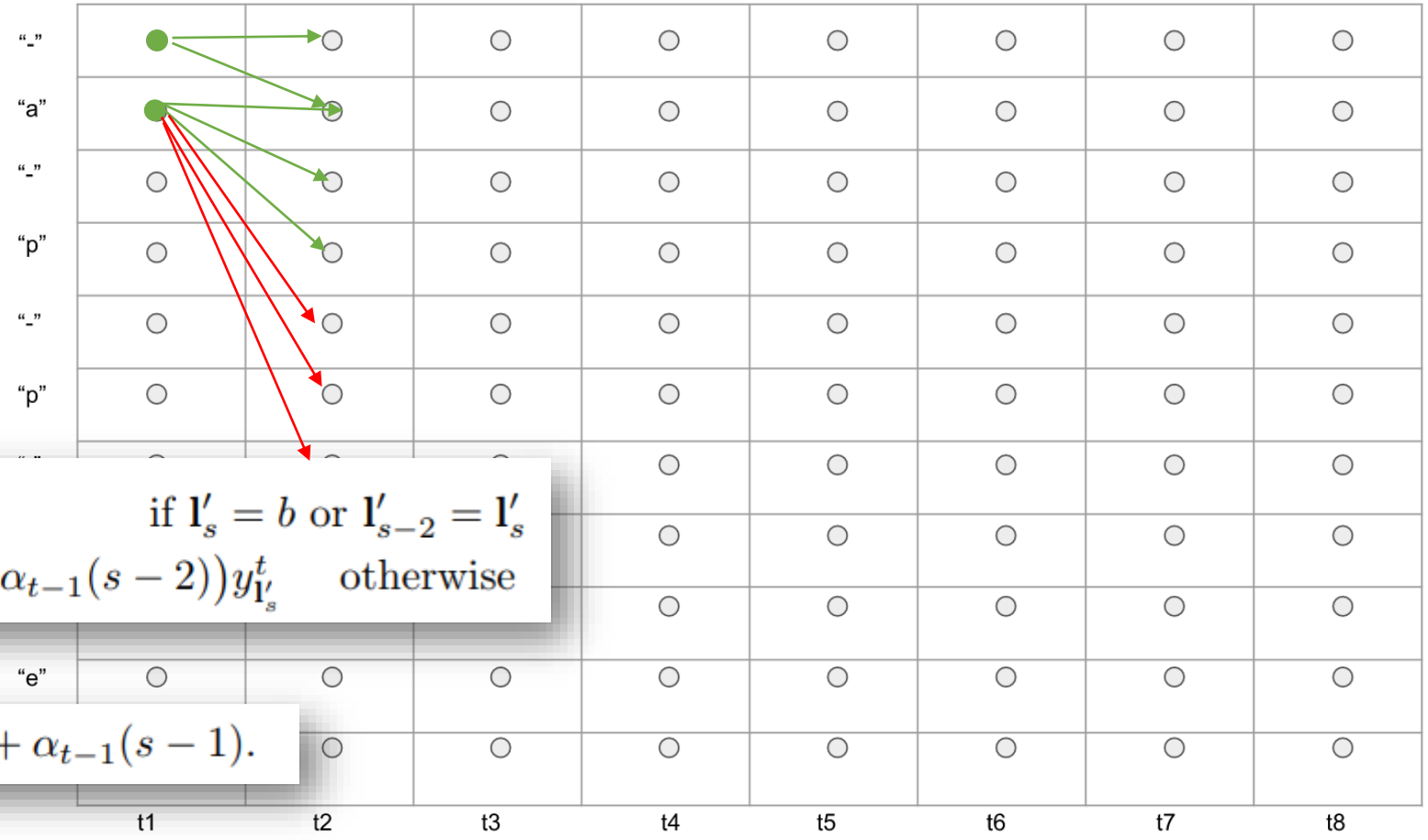


$$\alpha_t(s) = \begin{cases} \bar{\alpha}_t(s) y^t_{\mathbf{l}'_s} & \text{if } \mathbf{l}'_s = b \text{ or } \mathbf{l}'_{s-2} = \mathbf{l}'_s \\ \big(\bar{\alpha}_t(s) + \alpha_{t-1}(s-2)\big) y^t_{\mathbf{l}'_s} & \text{otherwise} \end{cases}$$

$$\bar{\alpha}_t(s) \overset{\text{def}}{=} \alpha_{t-1}(s) + \alpha_{t-1}(s-1).$$

Нахождение возможных путей

Куда можно двигаться из этих точек

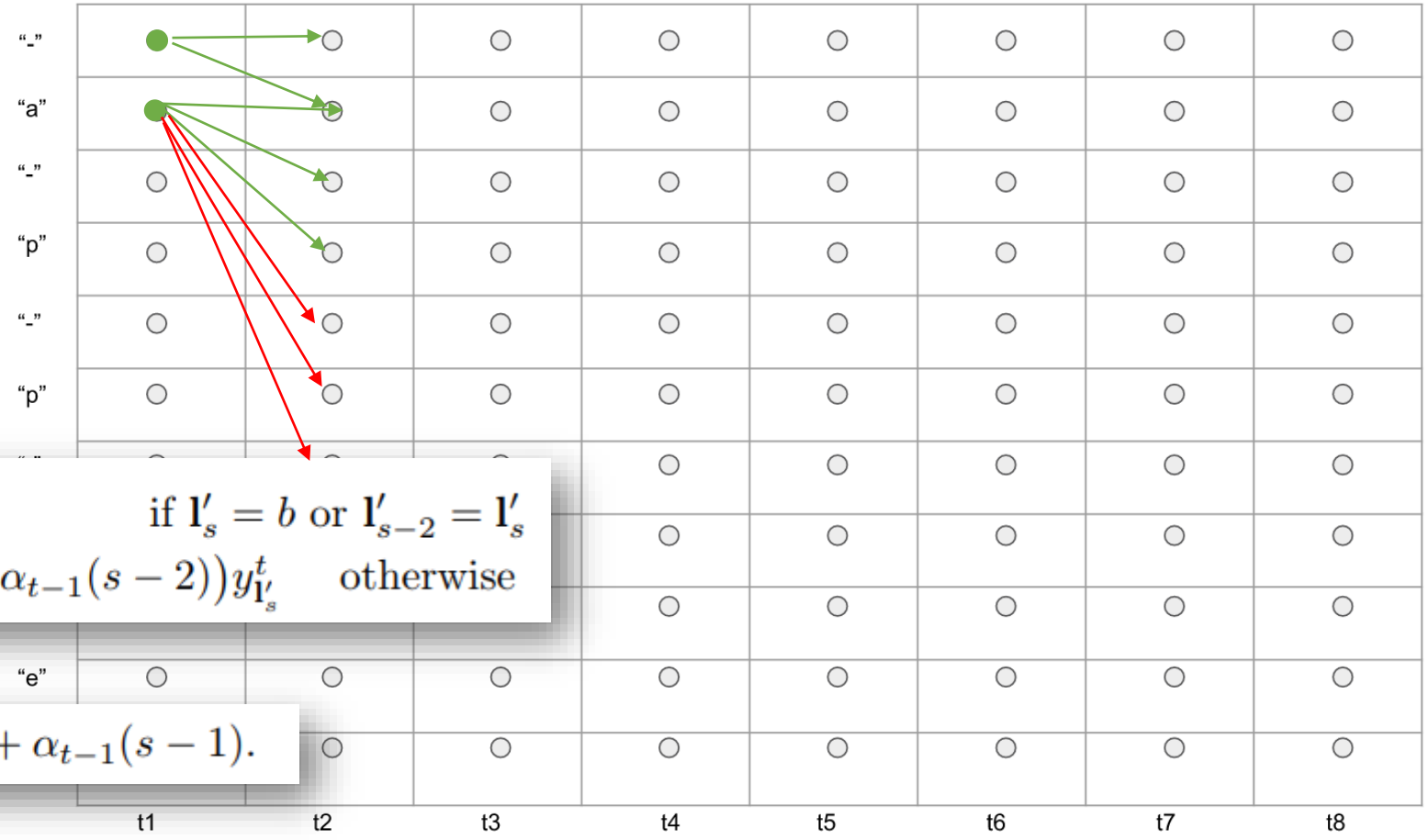


$$\alpha_t(s) = \begin{cases} \bar{\alpha}_t(s) y^t_{\mathbf{l}'_s} & \text{if } \mathbf{l}'_s = b \text{ or } \mathbf{l}'_{s-2} = \mathbf{l}'_s \\ \big(\bar{\alpha}_t(s) + \alpha_{t-1}(s-2)\big) y^t_{\mathbf{l}'_s} & \text{otherwise} \end{cases}$$

$$\bar{\alpha}_t(s) \overset{\text{def}}{=} \alpha_{t-1}(s) + \alpha_{t-1}(s-1).$$

Нахождение возможных путей

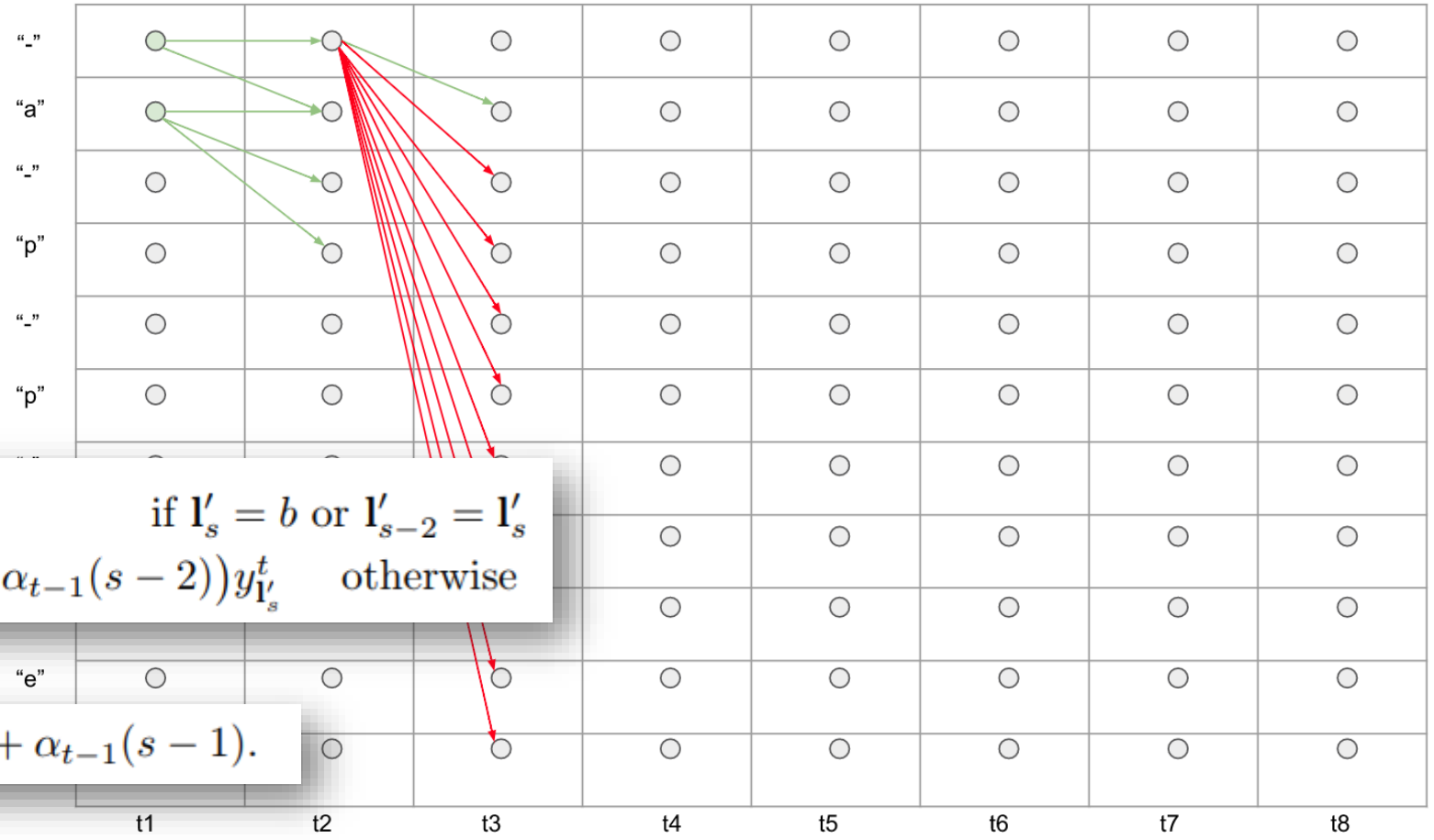
Куда можно двигаться из этих точек



$$\alpha_t(s) = \begin{cases} \bar{\alpha}_t(s) y^t_{\mathbf{l}'_s} & \text{if } \mathbf{l}'_s = b \text{ or } \mathbf{l}'_{s-2} = \mathbf{l}'_s \\ \big(\bar{\alpha}_t(s) + \alpha_{t-1}(s-2)\big) y^t_{\mathbf{l}'_s} & \text{otherwise} \end{cases}$$

$$\bar{\alpha}_t(s) \overset{\text{def}}{=} \alpha_{t-1}(s) + \alpha_{t-1}(s-1).$$

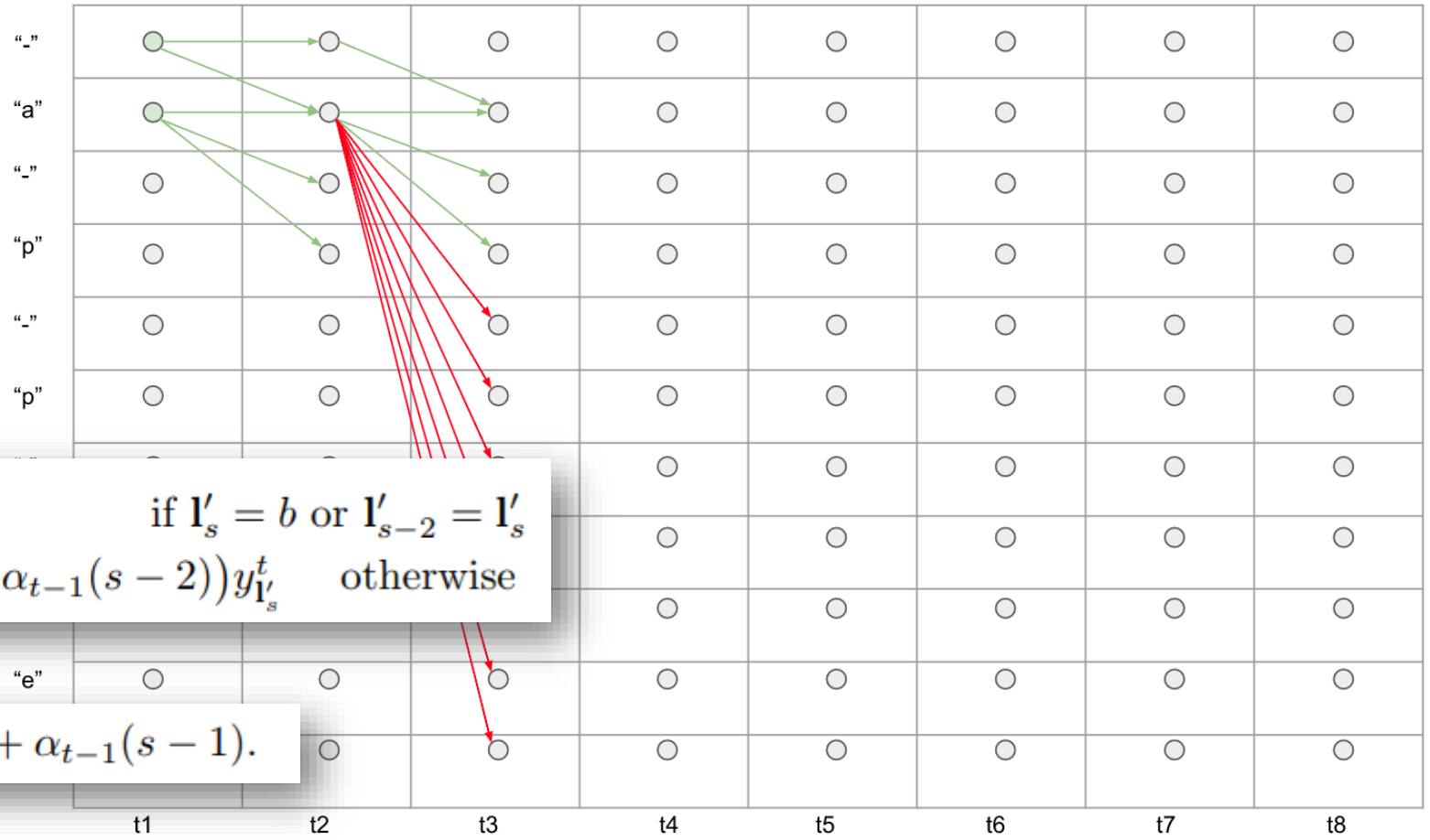
Нахождение возможных путей



$$\alpha_t(s) = \begin{cases} \bar{\alpha}_t(s) y^t_{\mathbf{l}'_s} & \text{if } \mathbf{l}'_s = b \text{ or } \mathbf{l}'_{s-2} = \mathbf{l}'_s \\ \big(\bar{\alpha}_t(s) + \alpha_{t-1}(s-2)\big) y^t_{\mathbf{l}'_s} & \text{otherwise} \end{cases}$$

$$\bar{\alpha}_t(s) \overset{\text{def}}{=} \alpha_{t-1}(s) + \alpha_{t-1}(s-1).$$

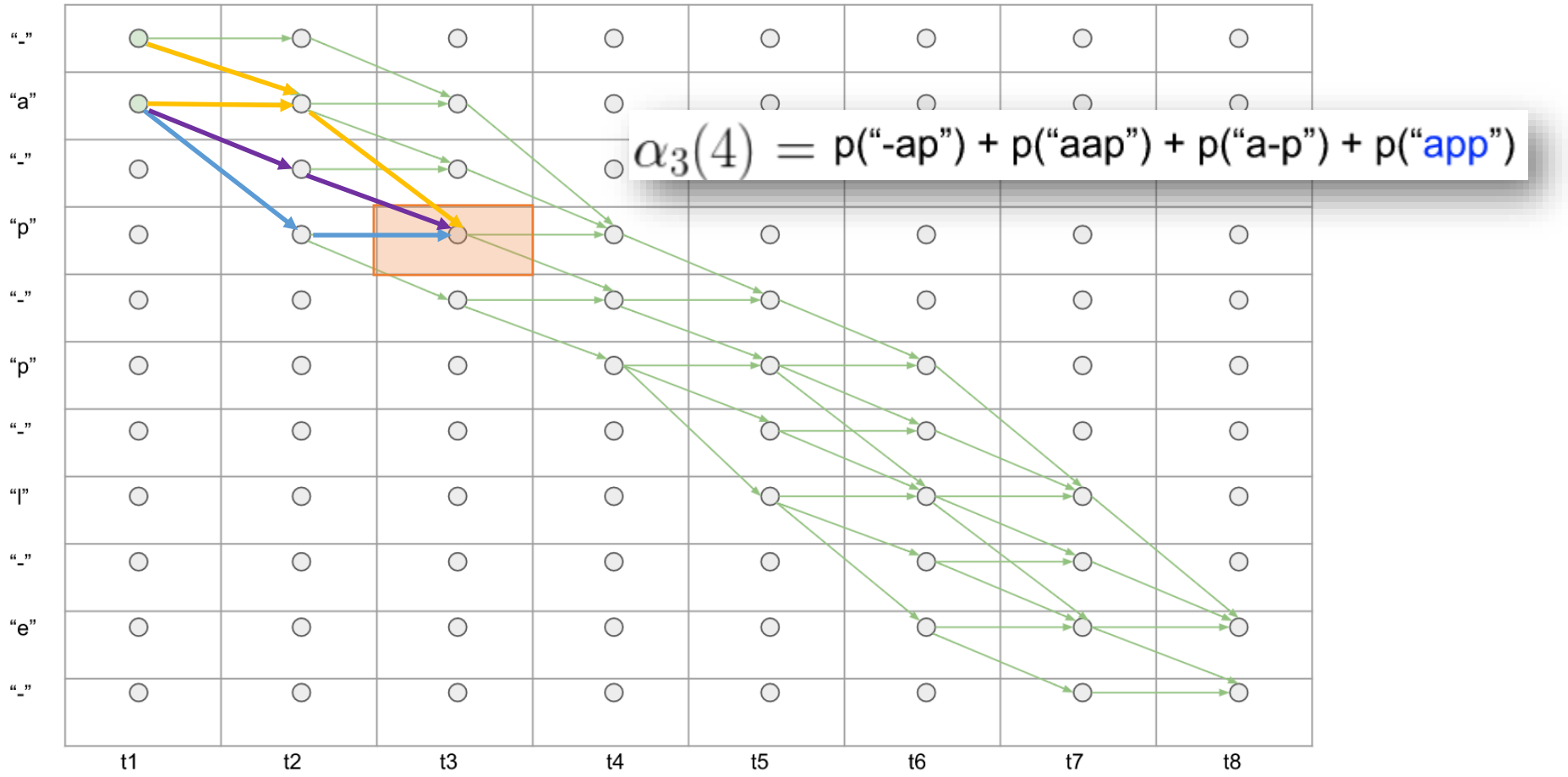
Нахождение возможных путей



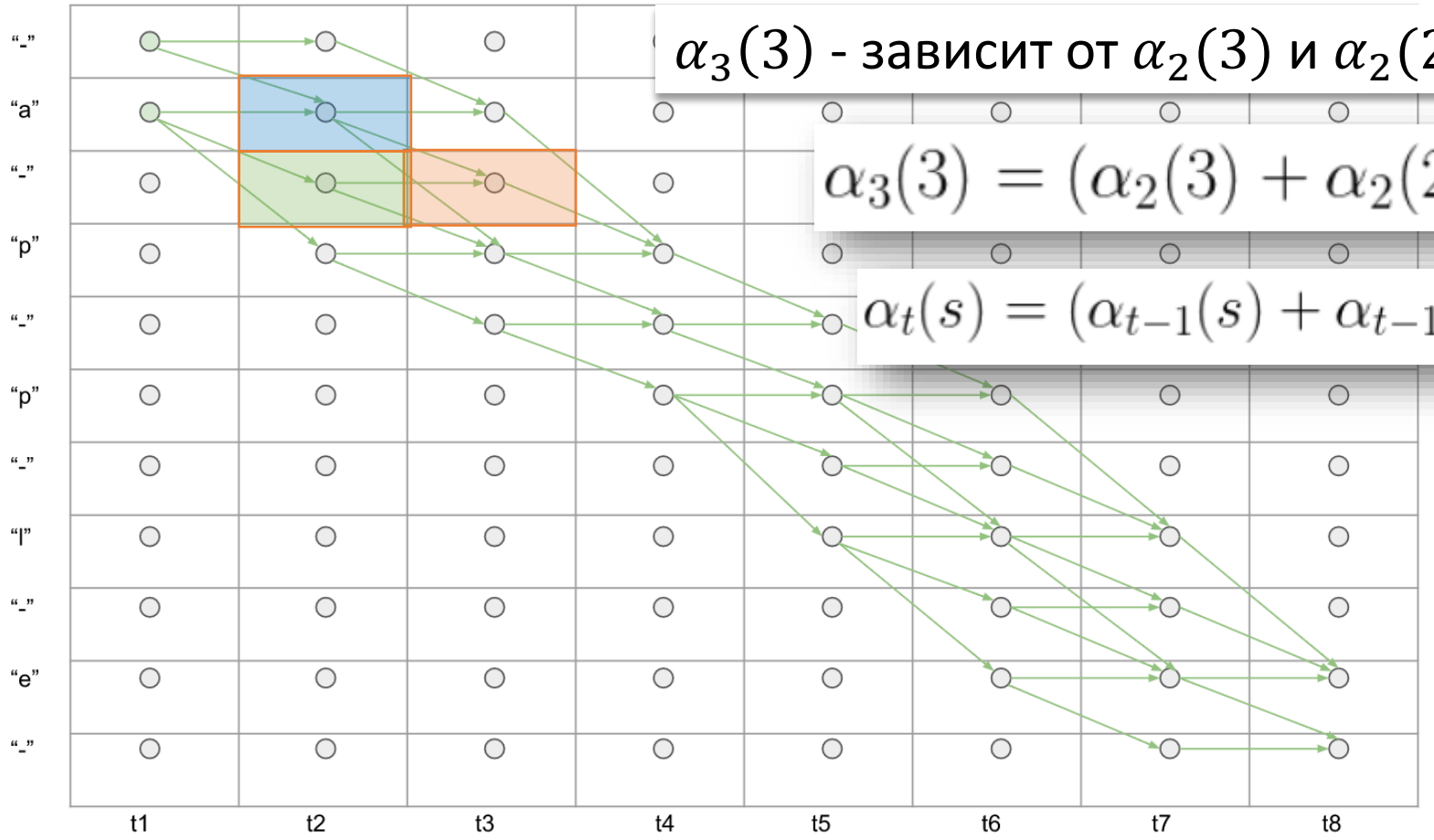
$$\alpha_t(s) = \begin{cases} \bar{\alpha}_t(s) y^t_{\mathbf{l}'_s} & \text{if } \mathbf{l}'_s = b \text{ or } \mathbf{l}'_{s-2} = \mathbf{l}'_s \\ \big(\bar{\alpha}_t(s) + \alpha_{t-1}(s-2)\big) y^t_{\mathbf{l}'_s} & \text{otherwise} \end{cases}$$

$$\bar{\alpha}_t(s) \overset{\text{def}}{=} \alpha_{t-1}(s) + \alpha_{t-1}(s-1).$$

Нахождение возможных путей



Нахождение возможных путей

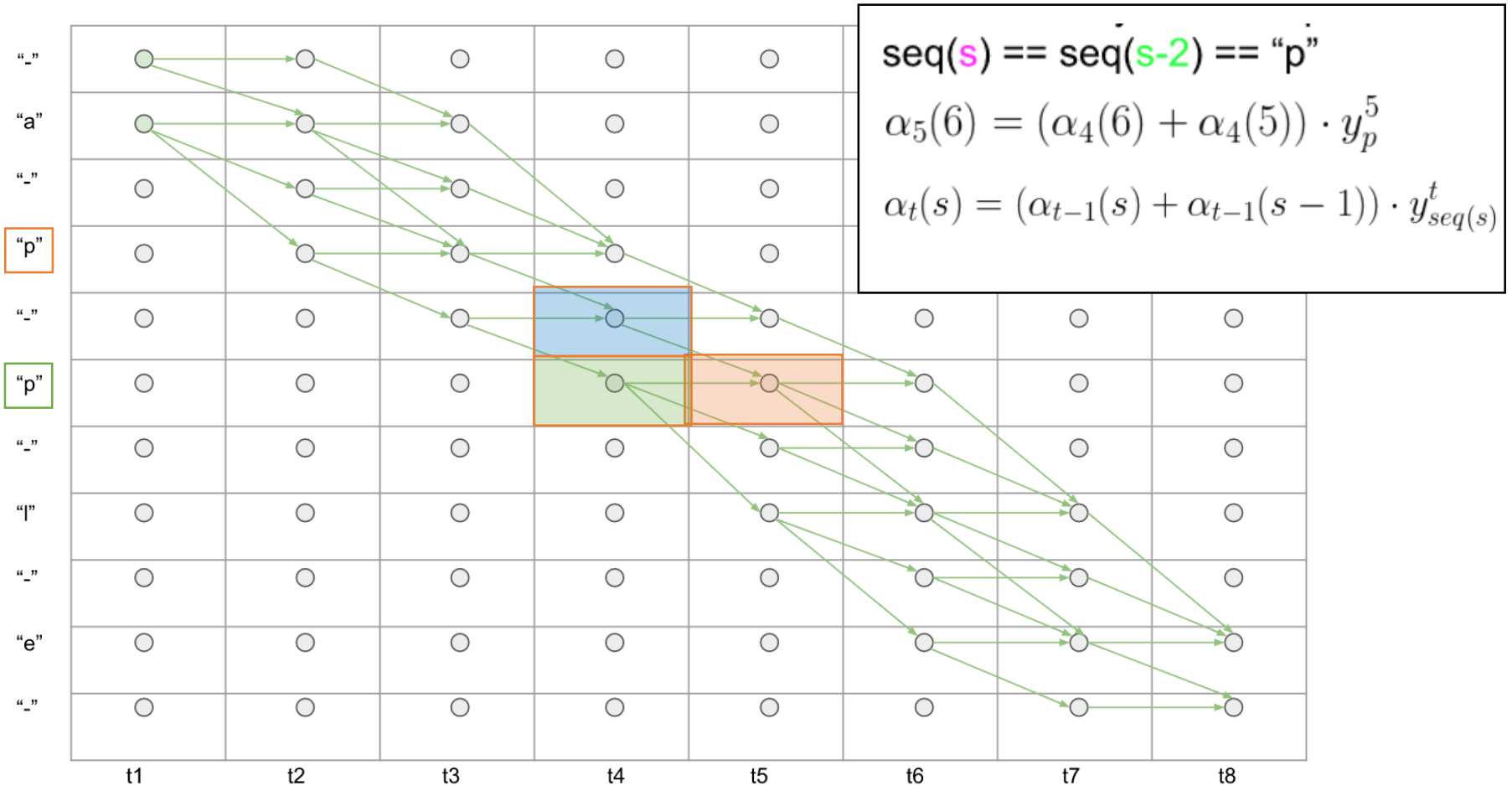


$\alpha_3(3)$ - зависит от $\alpha_2(3)$ и $\alpha_2(2)$

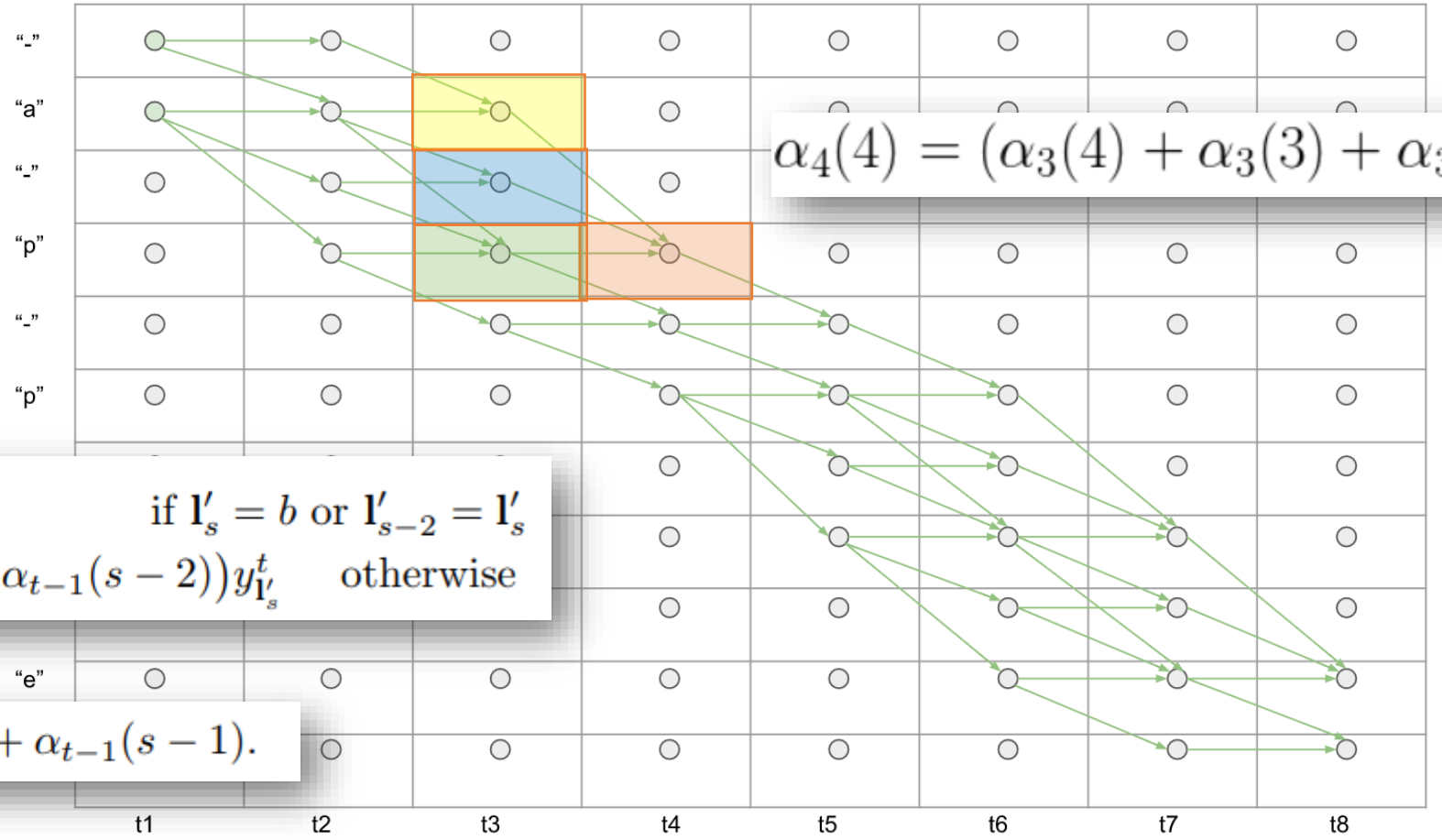
$$\alpha_3(3) = (\alpha_2(3) + \alpha_2(2)) \cdot y_-^3$$

$$\alpha_t(s) = (\alpha_{t-1}(s) + \alpha_{t-1}(s-1)) \cdot y_{seq(s)}^t$$

Нахождение возможных путей



Нахождение возможных путей

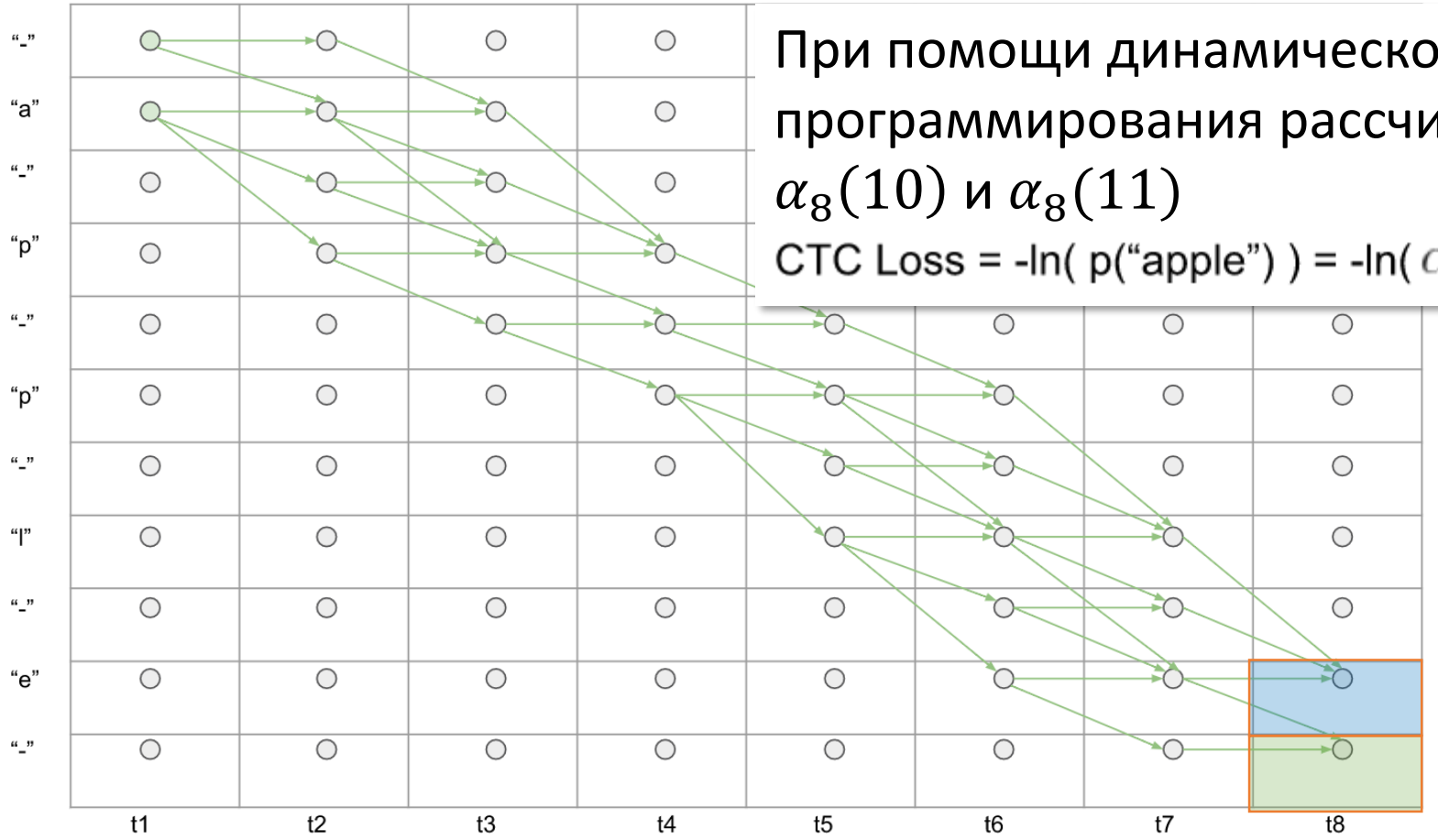


$$\alpha_4(4) = (\alpha_3(4) + \alpha_3(3) + \alpha_3(2)) \cdot y_p^4$$

$$\alpha_t(s) = \begin{cases} \bar{\alpha}_t(s) y^t_{\mathbf{l}'_s} & \text{if } \mathbf{l}'_s = b \text{ or } \mathbf{l}'_{s-2} = \mathbf{l}'_s \\ \big(\bar{\alpha}_t(s) + \alpha_{t-1}(s-2)\big) y^t_{\mathbf{l}'_s} & \text{otherwise} \end{cases}$$

$$\bar{\alpha}_t(s) \overset{\text{def}}{=} \alpha_{t-1}(s) + \alpha_{t-1}(s-1).$$

Нахождение возможных путей



При помощи динамического программирования рассчитываем $\alpha_8(10)$ и $\alpha_8(11)$

CTC Loss = -ln(p(“apple”)) = -ln($\alpha_8(10)$ + $\alpha_8(11)$)

Нахождение возможных путей. Обратный проход

- Делается аналогично прямому проходу для каждого момента времени и метки рассчитываем вероятность

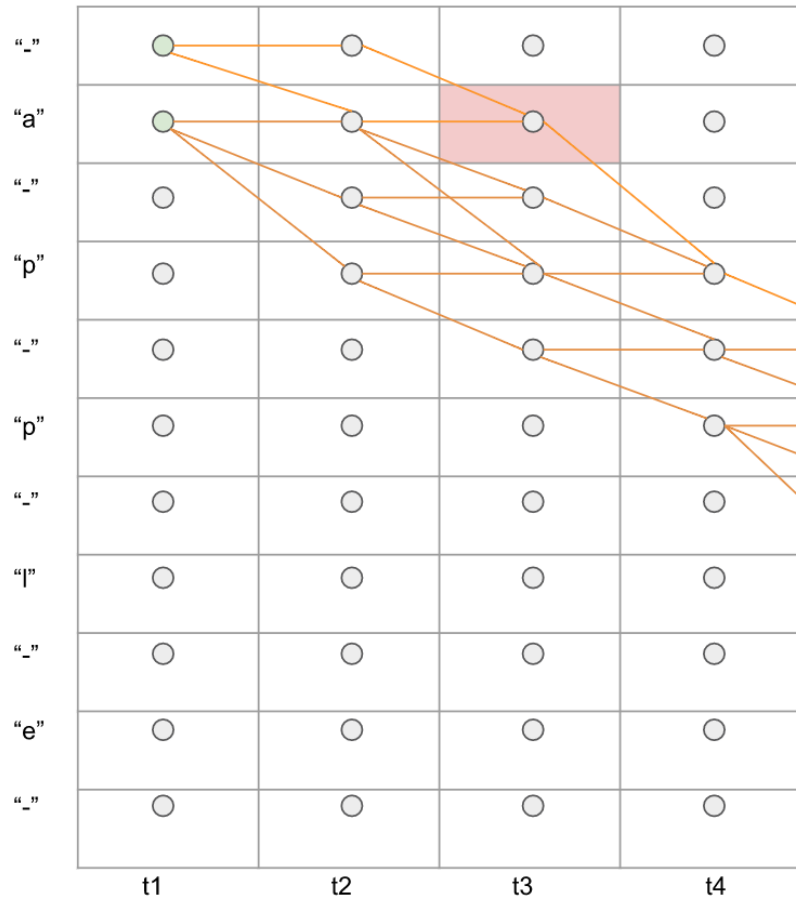
$$\beta_t(s) \stackrel{\text{def}}{=} \sum_{\substack{\pi \in N^T: \\ \mathcal{B}(\pi_{t:T}) = \mathbf{l}_{s:|\mathbf{l}|}}} \prod_{t'=t}^{T} y^{t'}_{\pi_{t'}}$$

$$\begin{aligned} \beta_T(|\mathbf{l}'|) &= y^T_b \\ \beta_T(|\mathbf{l}'| - 1) &= y^T_{\mathbf{l}_{|\mathbf{l}|}} \\ \beta_T(s) &= 0, \ \forall s < |\mathbf{l}'| - 1 \end{aligned}$$

$$\beta_t(s) = \begin{cases} \bar{\beta}_t(s) y^t_{\mathbf{l}'_s} & \text{if } \mathbf{l}'_s = b \text{ or } \mathbf{l}'_{s+2} = \mathbf{l}'_s \\ \big(\bar{\beta}_t(s) + \beta_{t+1}(s+2)\big) y^t_{\mathbf{l}'_s} & \text{otherwise} \end{cases}$$

$$\bar{\beta}_t(s) \stackrel{\text{def}}{=} \beta_{t+1}(s) + \beta_{t+1}(s+1).$$

Вероятность путей для отдельной метки



$\alpha_3(2)$ = p(“--a”) + p(“-aa”) + p(“aaa”) =

$$= y_-^1 \cdot y_-^2 \cdot y_a^3 + y_-^1 \cdot y_a^2 \cdot y_a^3 + y_a^1 \cdot y_a^2 \cdot y_a^3$$

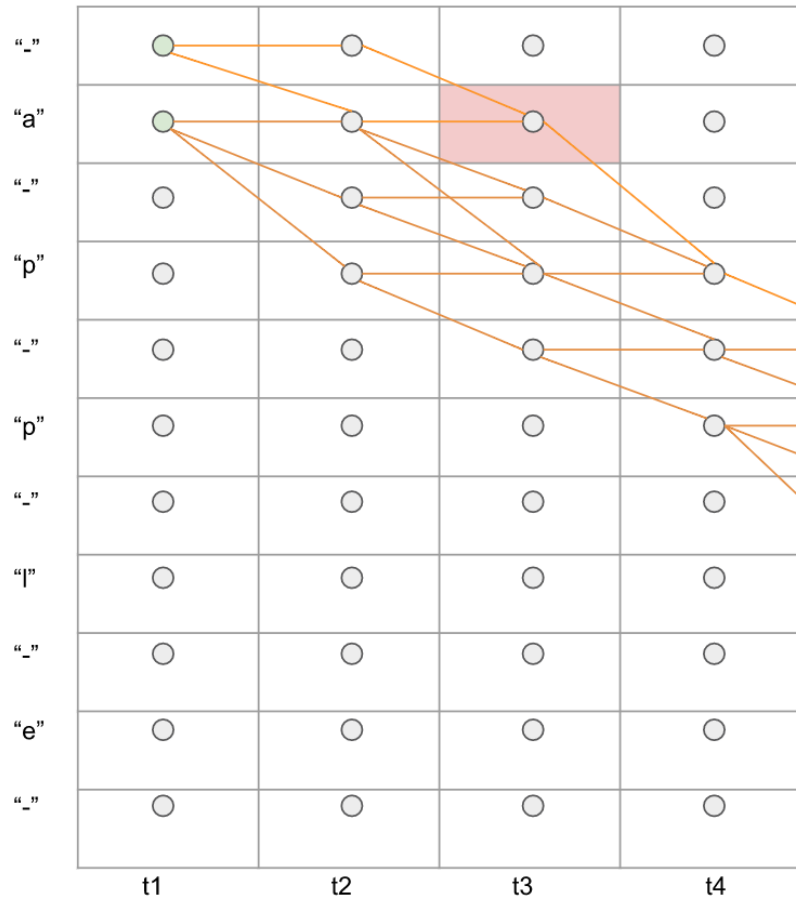
$\beta_3(2)$= p(“ap-ple”)= $y_a^3 \cdot y_p^4 \cdot y_-^5 \cdot y_p^6 \cdot y_l^7 \cdot y_e^8$

=> $\alpha_3(2) \cdot \beta_3(2) = y_-^1 \cdot y_-^2 \cdot y_a^3 \cdot y_a^3 \cdot y_p^4 \cdot y_-^5 \cdot y_p^6 \cdot y_l^7 \cdot y_e^8$ +

+ $y_-^1 \cdot y_a^2 \cdot y_a^3 \cdot y_a^3 \cdot y_p^4 \cdot y_-^5 \cdot y_p^6 \cdot y_l^7 \cdot y_e^8$

+ $y_a^1 \cdot y_a^2 \cdot y_a^3 \cdot y_a^3 \cdot y_p^4 \cdot y_-^5 \cdot y_p^6 \cdot y_l^7 \cdot y_e^8$

Вероятность путей для отдельной метки



$$\alpha_3(2) \cdot \beta_3(2) = y_-^1 \cdot y_-^2 \cdot y_a^3 \cdot \boxed{y_a^3} \cdot y_p^4 \cdot y_-^5 \cdot y_p^6 \cdot y_l^7 \cdot y_e^8 \quad +$$

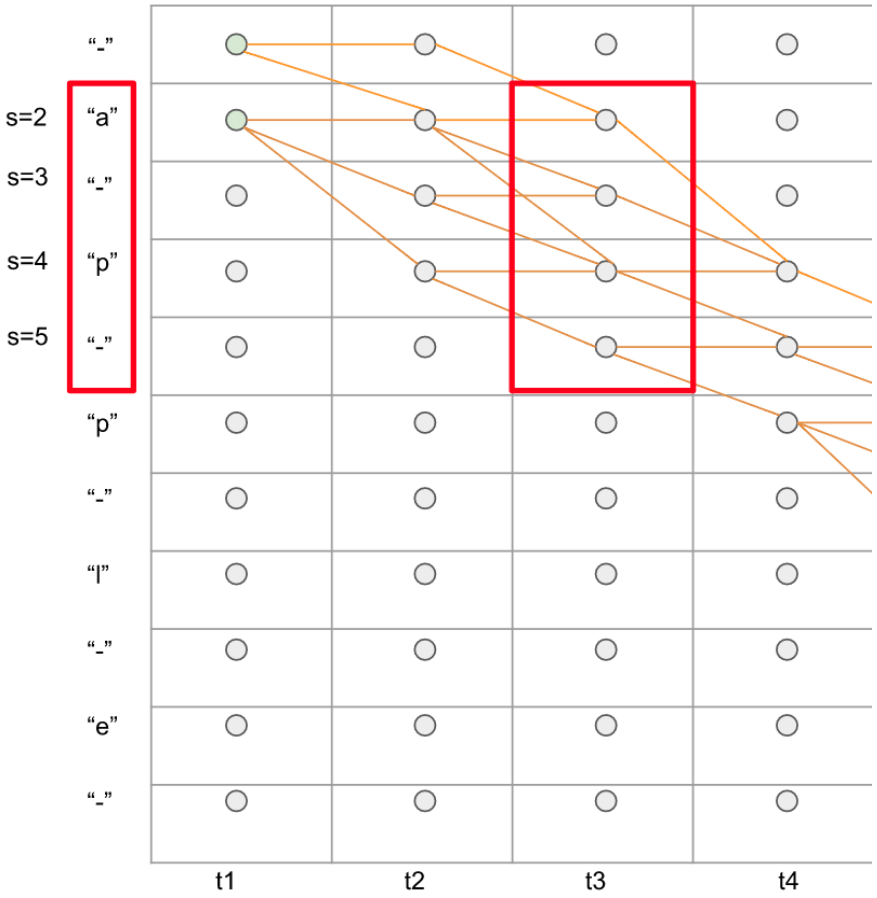
$$+ \quad y_-^1 \cdot y_a^2 \cdot y_a^3 \cdot \boxed{y_a^3} \cdot y_p^4 \cdot y_-^5 \cdot y_p^6 \cdot y_l^7 \cdot y_e^8$$

$$+ \quad y_a^1 \cdot y_a^2 \cdot y_a^3 \cdot \boxed{y_a^3} \cdot y_p^4 \cdot y_-^5 \cdot y_p^6 \cdot y_l^7 \cdot y_e^8 =$$

$$= (\ p(\text{“--aap-ple”}) + p(\text{“-aap-ple”}) + p(\text{“aaap-ple”})\ ) * y_a^3$$

$$\frac{\alpha_3(2) \cdot \beta_3(2)}{y_a^3} = (\ p(\text{“--aap-ple”}) + p(\text{“-aap-ple”}) + p(\text{“aaap-ple”})\ )$$

Вероятность слова для времени



Суммарная вероятность всех путей на шаге 3

$$p(\text{"apple"}) = \sum_{s=2}^{5} \frac{\alpha_3(s) \cdot \beta_3(s)}{y^3_{seq(s)}}$$

Вероятность в любой момент времени

$$p(\text{"apple"}) = \sum_{s=1}^{|seq|} \frac{\alpha_t(s) \cdot \beta_t(s)}{y^t_{seq(s)}}$$

CTC Loss = -ln(p(“apple”))

Градиент

$$p(\text{"}apple\text{"}) = \sum_{s=1}^{|seq|} \frac{\alpha_t(s) \cdot \beta_t(s)}{y^t_{seq(s)}}$$

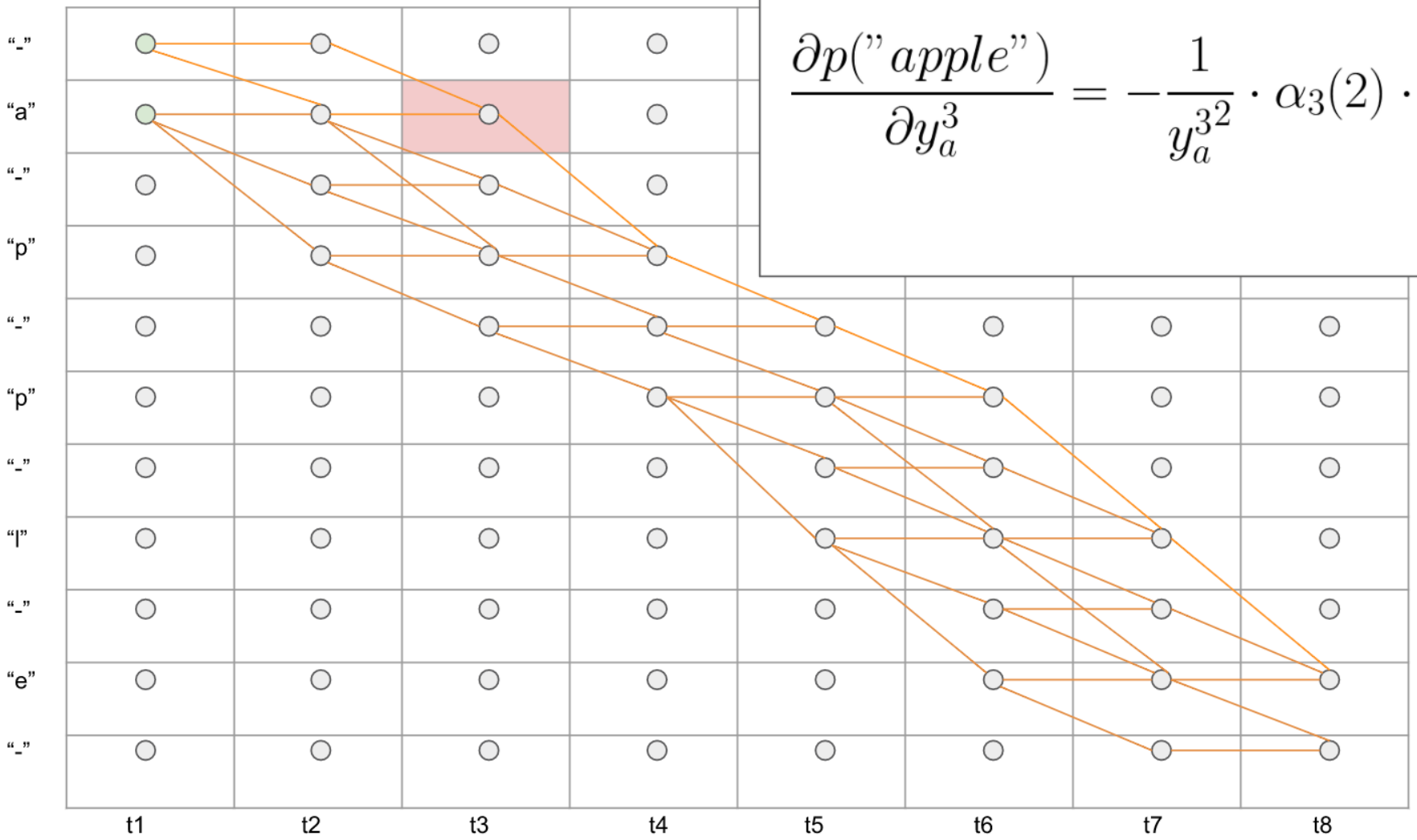
- Принимаем во внимание, что

$$\frac{\partial(-ln(p(\text{"}apple\text{"})))}{\partial y^t_k} = -\frac{1}{p(\text{"}apple\text{"})} \cdot \frac{\partial p(\text{"}apple\text{"})}{\partial y^t_k}$$

- Чтобы обучить сеть нам нужно продифференцировать функцию ошибки по всем выходам сети y^t_k
- Рассматриваем только пути идущие через символ k в момент t

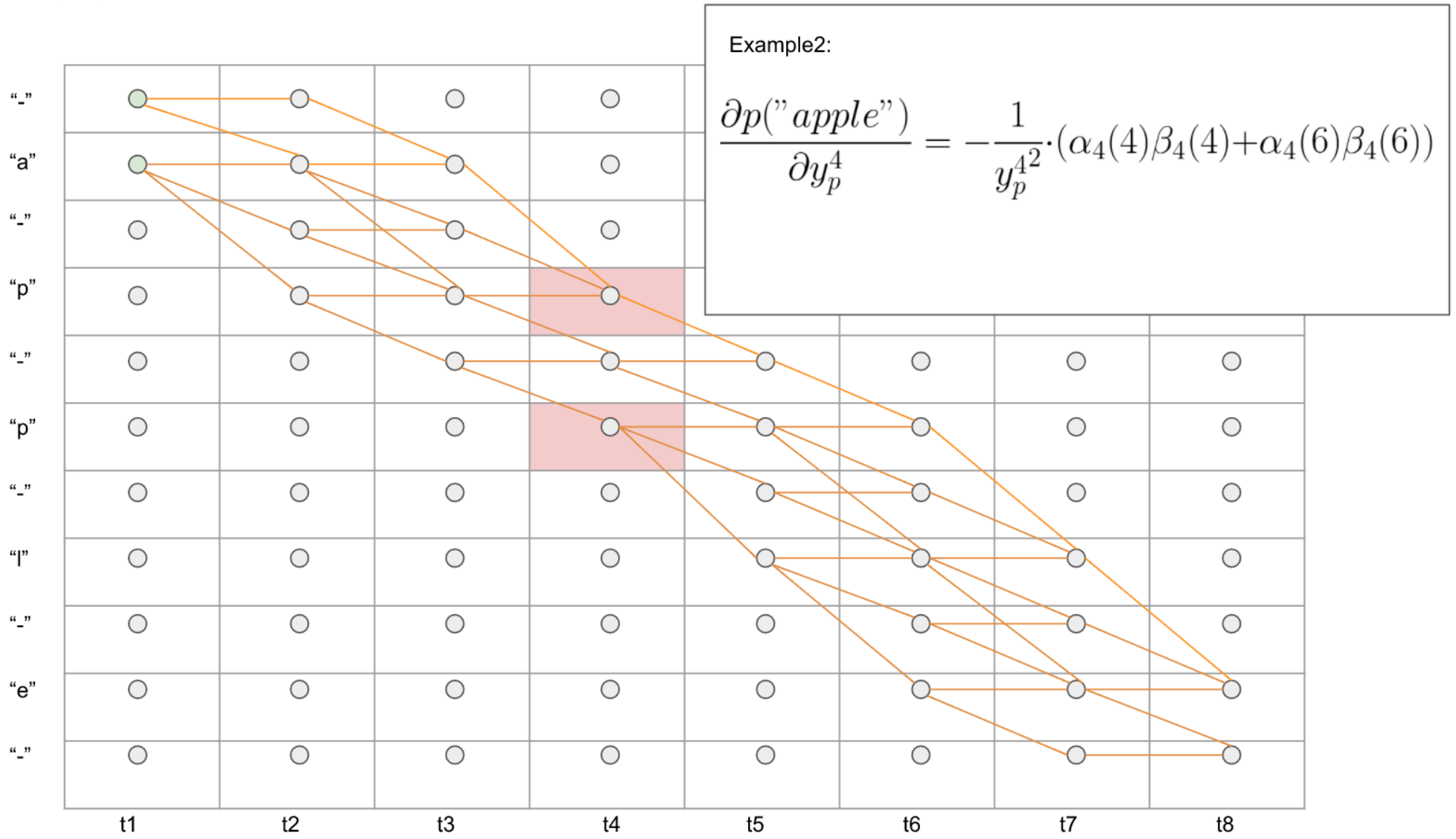
$$\frac{\partial p(\text{"}apple\text{"})}{\partial y^t_k} = -\frac{1}{{y^t_k}^2} \cdot \sum_{s:seq(s)=k} \alpha_t(s) \cdot \beta_t(s)$$

Градиент



$$\frac{\partial p(\text{"apple"})}{\partial y_a^3} = -\frac{1}{y_a^{3^2}} \cdot \alpha_3(2) \cdot \beta_3(2)$$

Градиент


Example2:
$\frac{\partial p(\text{"apple"})}{\partial y_p^4} = -\frac{1}{y_p^{4^2}} \cdot (\alpha_4(4)\beta_4(4) + \alpha_4(6)\beta_4(6))$
"-"
"a"
"-"
"p"
"-"
"p"
"-"
"l"
"-"
"e"
"-"
t1
t2
t3
t4
t5
t6
t7
t8

Deep Speech

- Вход – транскрибированное аудио самплы (частота дискретизации 8К)

$$\mathcal{X} = \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots\}.$$

- Фичи – спектрограммы. 80 линейно распределенных логарифмических фильтр-банков.

Deep Speech

- Архитектура сети

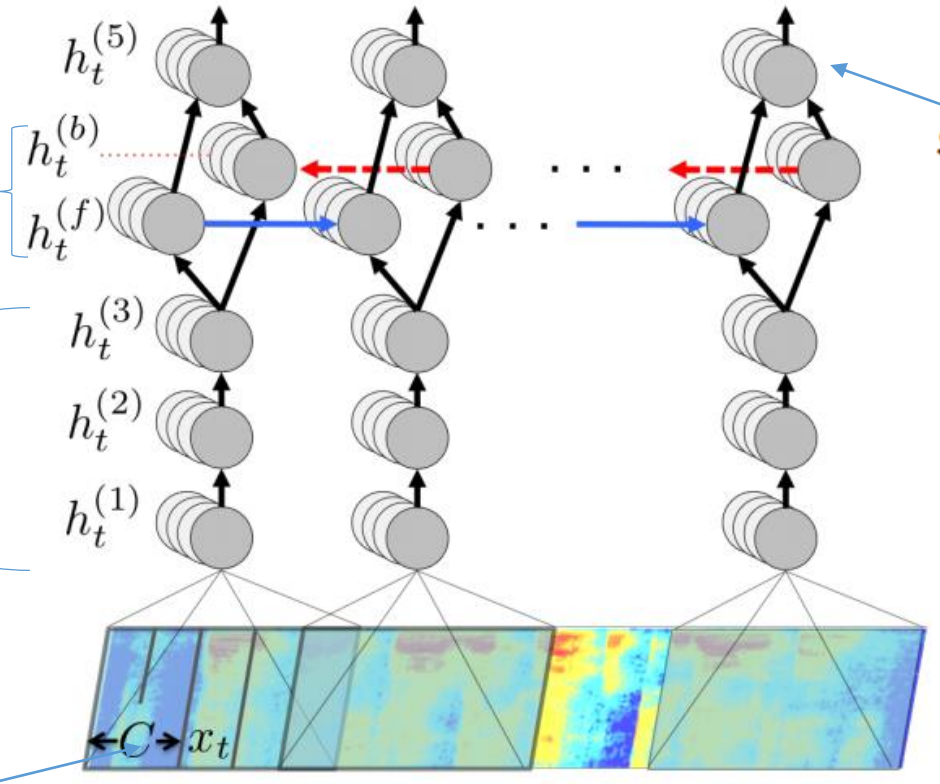
$$h_{t,k}^{(6)} = \hat{y}_{t,k} \equiv \mathbb{P}(c_t = k|x) = \frac{\exp(W_k^{(6)} h_t^{(5)} + b_k^{(6)})}{\sum_j \exp(W_j^{(6)} h_t^{(5)} + b_j^{(6)})}.$$

Двунаправленная RNN

$$h_t^{(f)} = g(W^{(4)} h_t^{(3)} + W_r^{(f)} h_{t-1}^{(f)} + b^{(4)})$$
$$h_t^{(b)} = g(W^{(4)} h_t^{(3)} + W_r^{(b)} h_{t+1}^{(b)} + b^{(4)})$$

Линейный слой

$$h_t^{(l)} = g(W^{(l)} h_t^{(l-1)} + b^{(l)})$$



Линейный слой

$$g(W^{(5)} h_t^{(4)} + b^{(5)})$$

g = ReLu

Количество сгруппированных фреймов
C ∈ {5, 7, 9}

Deep Speech

- Предсказывает последовательность символов
- Loss = CTC
- Для предсказания использует языковую модель

$$Q(c) = \log(\mathbb{P}(c|x)) + \alpha \log(\mathbb{P}_{\text{lm}}(c)) + \beta \, \text{word\_count}(c)$$

Вероятность последователь ности RNN

Вероятность последовательности языковой модели

RNN output	Decoded Transcription
what is the weather like in bostin right now	what is the weather like in boston right now
prime miniter nerenr modi	prime minister narendra modi
arther n tickets for the game	are there any tickets for the game