# ОБУЧЕНИЕ ГЛУБОКИХ СЕТЕЙ VI

# ОБРАБОТКА ЕСТЕСТВЕННОГО ЯЗЫКА

Сергей Николенко

Вымпелком, Москва
14 июня 2017 г.

*Random facts*:

- 14 июня 1964 г. на Гаити проходил референдум; на бюллетенях для голосования был уже заранее напечатан ответ «Да» на вопрос о согласии с тем, чтобы Франсуа Дювалье стал пожизненным президентом
- 14 июня 1859 г. Чечня была присоединена к Российской империи, и через два месяца Шамиль был вынужден сдаться в плен; а 14 июня 1995 г. Шамиль Басаев захватил более 1600 заложников в больнице Будённовска
- 14 июня -- день памяти преподобного Агапита Печерского, врача безмездного, исцелявшего тяжелобольных молитвой и варёным былием

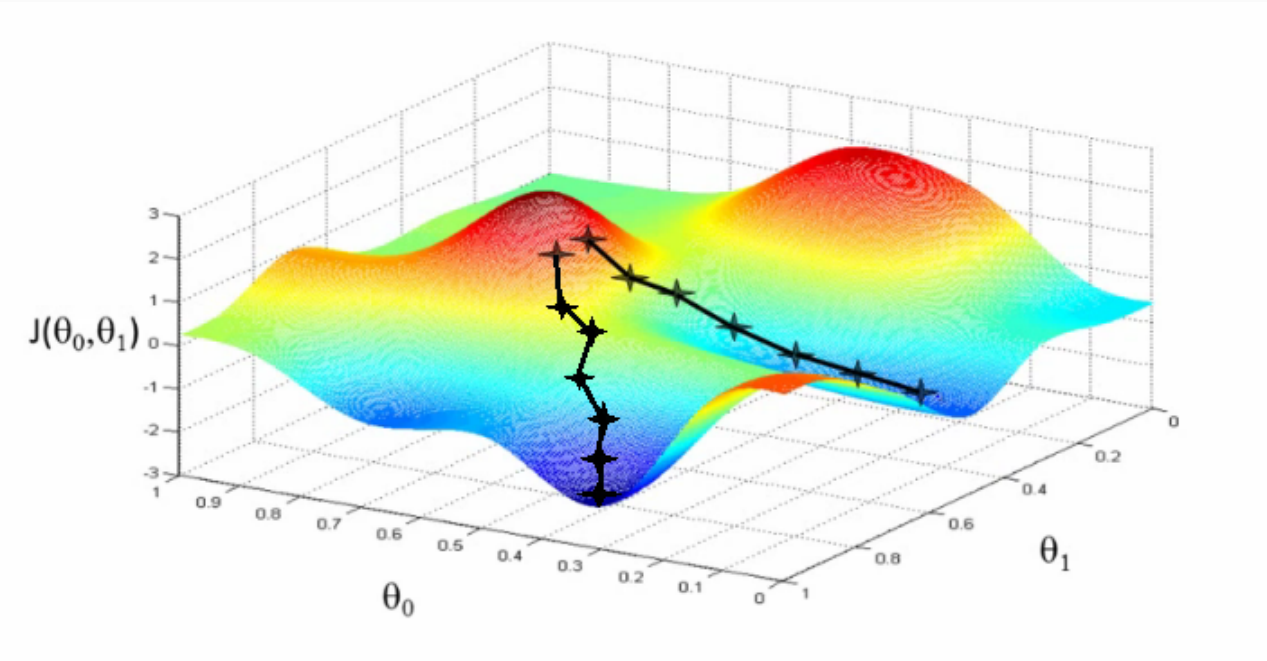
# КРАТКОЕ СОДЕРЖАНИЕ ПРЕДЫДУЩИХ СЕРИЙ

# Машинное обучение как оптимизация

- В машинном обучении много разных задач, с учителем и без.
- Их обычно решают по теореме Байеса, пересчитывая наши представления о параметрах в зависимости от полученных данных.
- Для этого обычно надо оптимизировать какую-нибудь функцию.
- Машинное обучение — это наука об аппроксимация и оптимизации функций (правдоподобие, апостериорное распределение, просто функция ошибки).
- И для невыпуклых функций это обычно делают тем или иным вариантом *градиентного спуска*.

# ГРАДИЕНТНЫЙ СПУСК

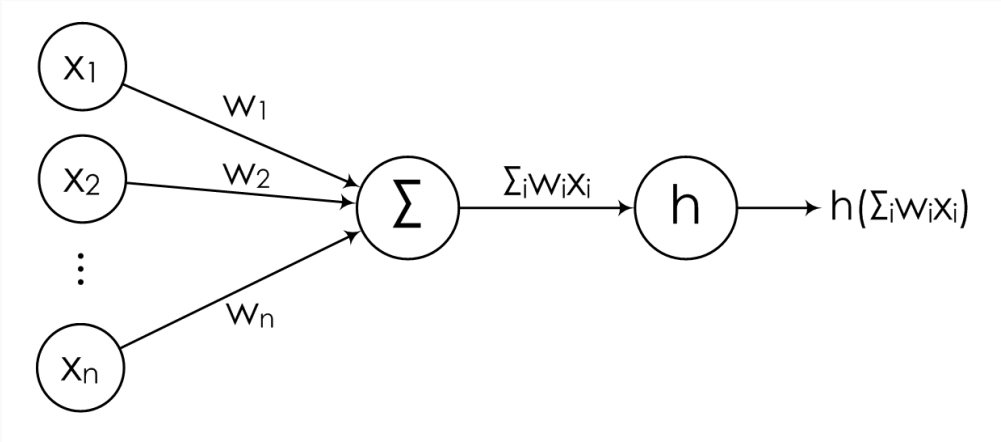
- Градиентный спуск — главный и фактически единственный способ оптимизации очень сложных функций.



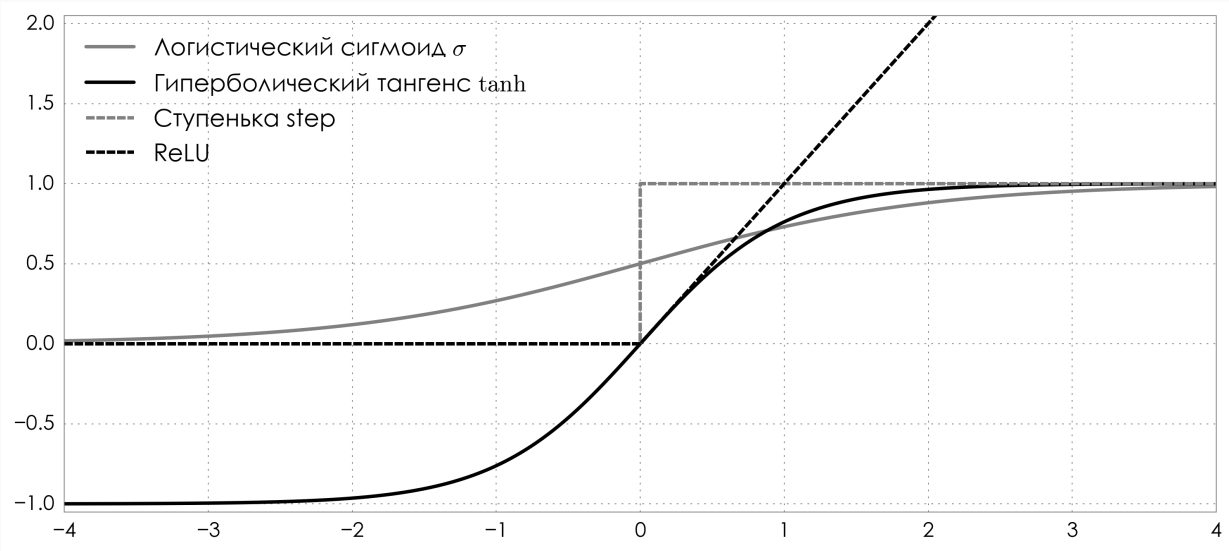

- Берём градиент $\nabla E(\mathbf{w})$, сдвигаемся немножко, повторяем.

# ПЕРЦЕПТРОН

- *Перцептрон*: $y = h(\mathbf{w}^\top \mathbf{x}) = h\left(\sum_i w_i x_i\right).$

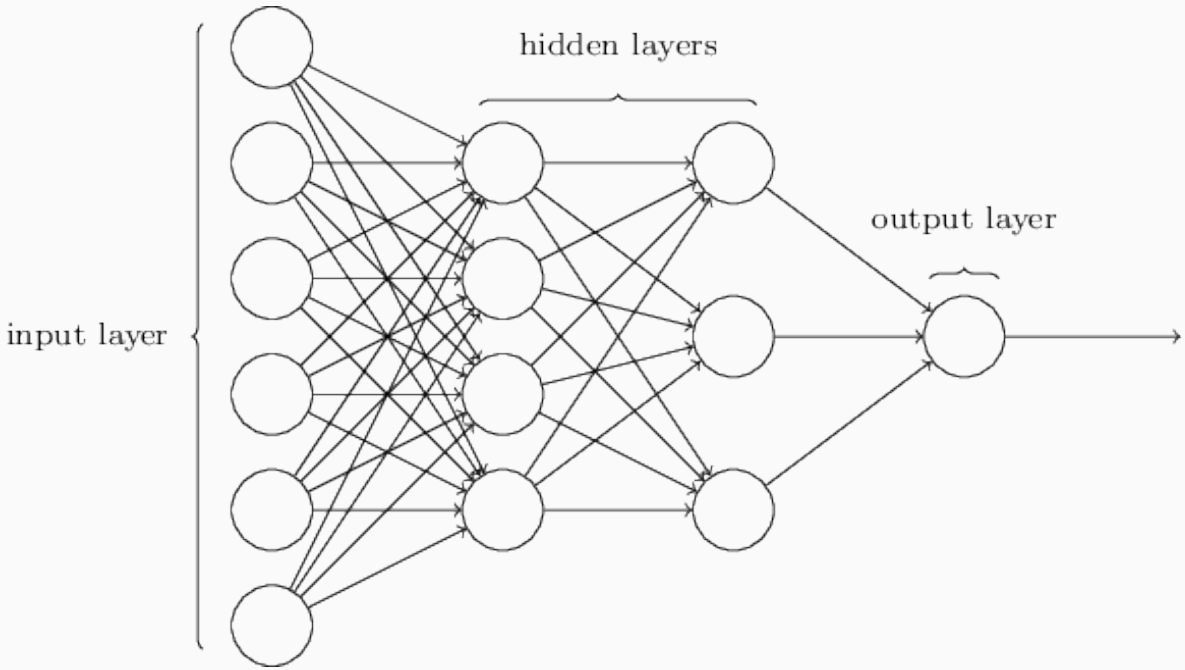



- Разные нелинейные функции:



# Объединяем перцептроны в сеть

- Сеть перцептронов; выходы одних – входы других.
- Hornik, 1990: двух уровней достаточно для приближения любой функции.
- Глубокие сети эффективнее — distributed representations.
- Обычно нейронные сети организованы в *слои*.

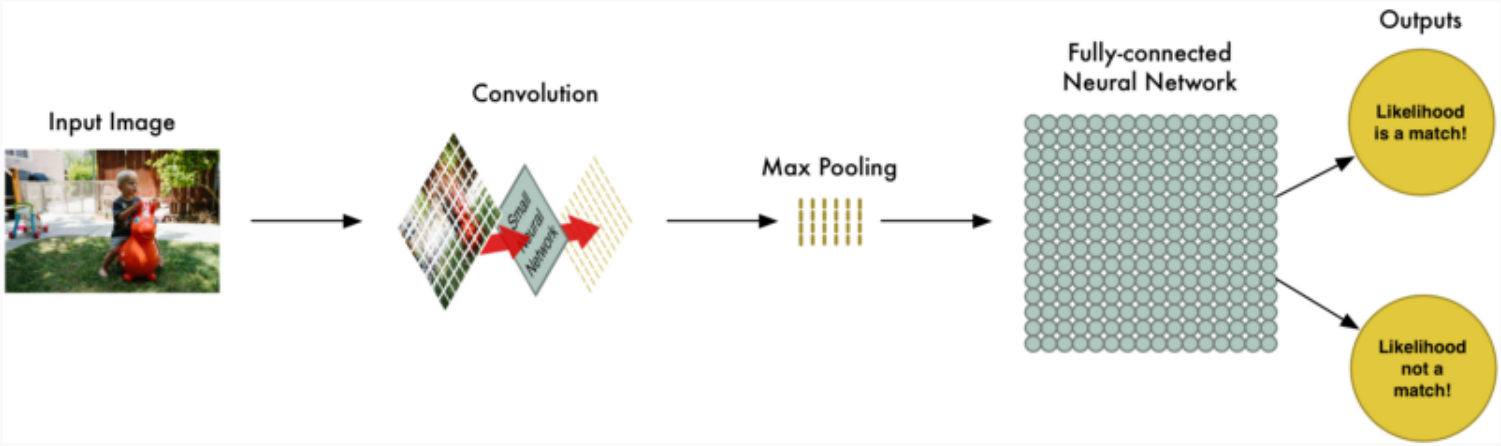


# Современный градиентный спуск

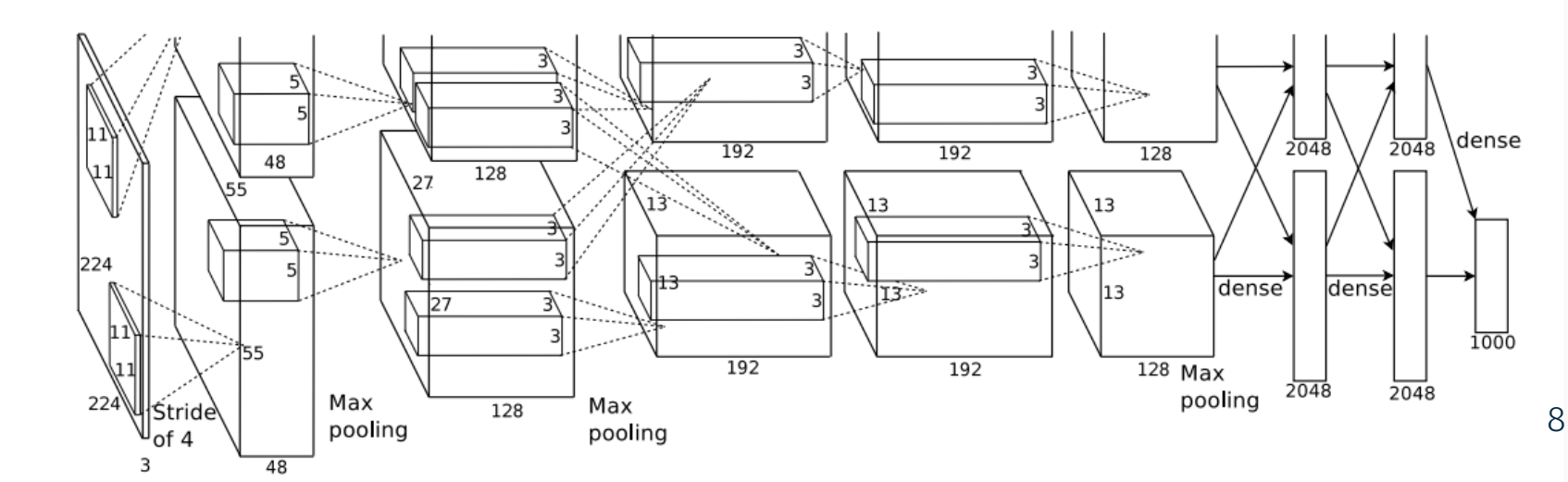
- В нейронной сети очень много параметров:
  - регуляризация ($L_2$, $L_1$, дропаут);
  - инициализация (предобучение без учителя, случайная);
  - нормализация (по мини-батчам, по уровням);
  - варианты градиентного спуска (моменты, Нестеров, Adadelta, Adam).

# СВЁРТОЧНЫЕ НЕЙРОННЫЕ СЕТИ

- Базовый свёрточный слой: свёртка, нелинейность, субдискретизация.

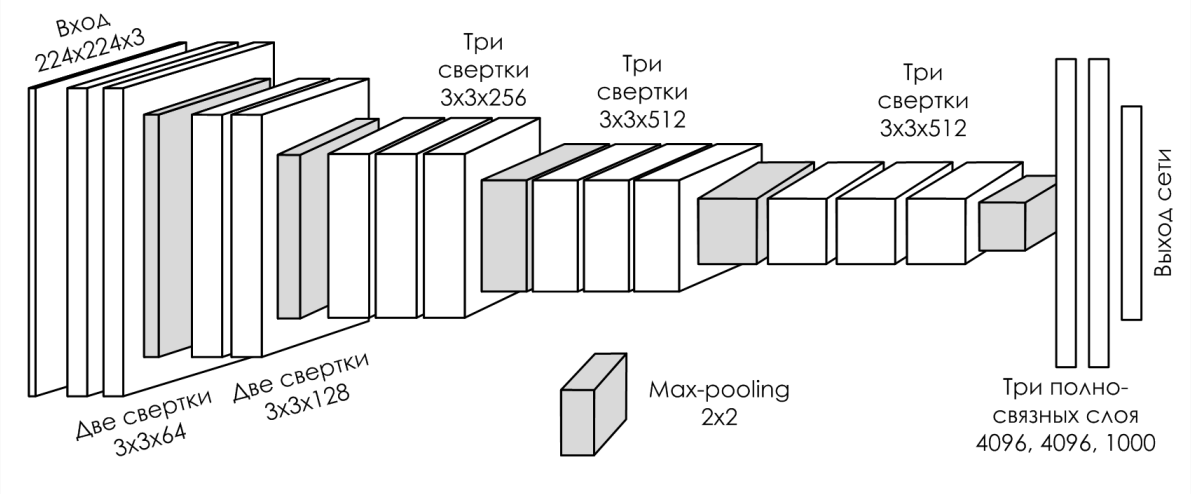



- Свёрточные сети могут быть глубокими (LeNet, AlexNet):

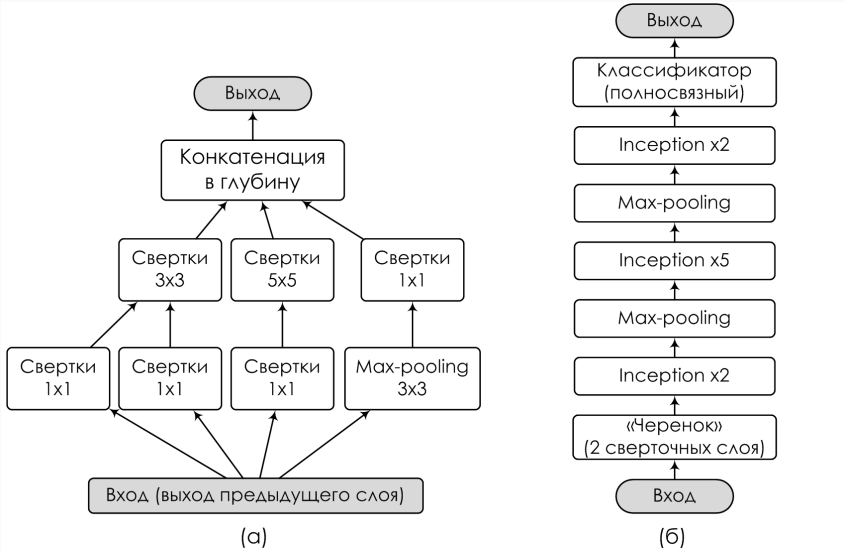


# СВЁРТОЧНЫЕ НЕЙРОННЫЕ СЕТИ

- VGG: достаточно брать $3 \times 3$ свёртки:

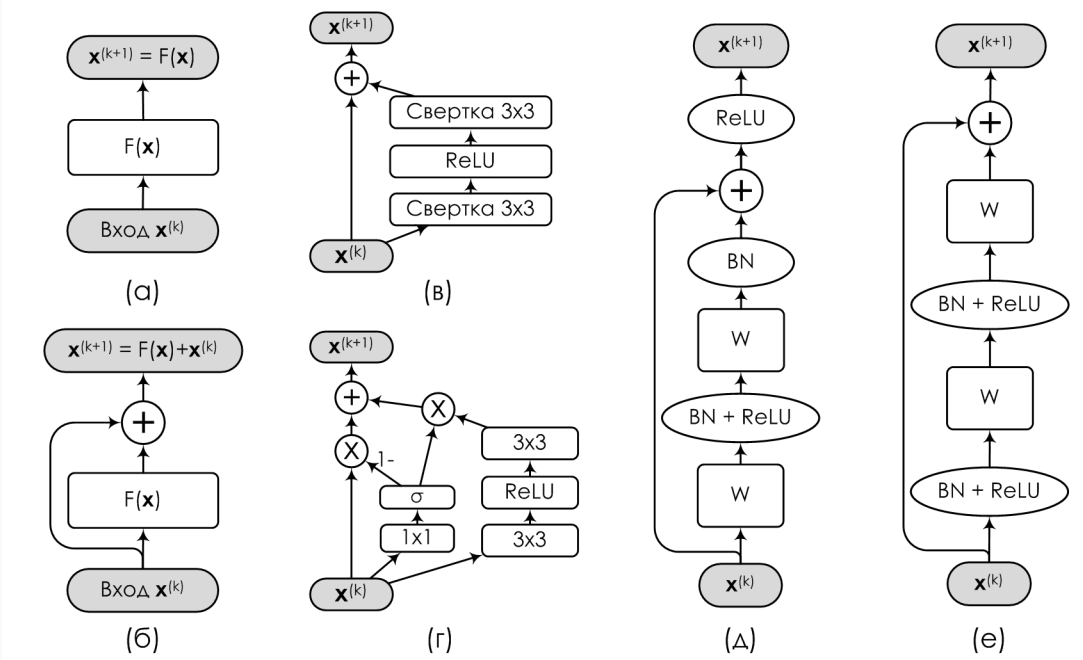



- Network in Network, Inception:

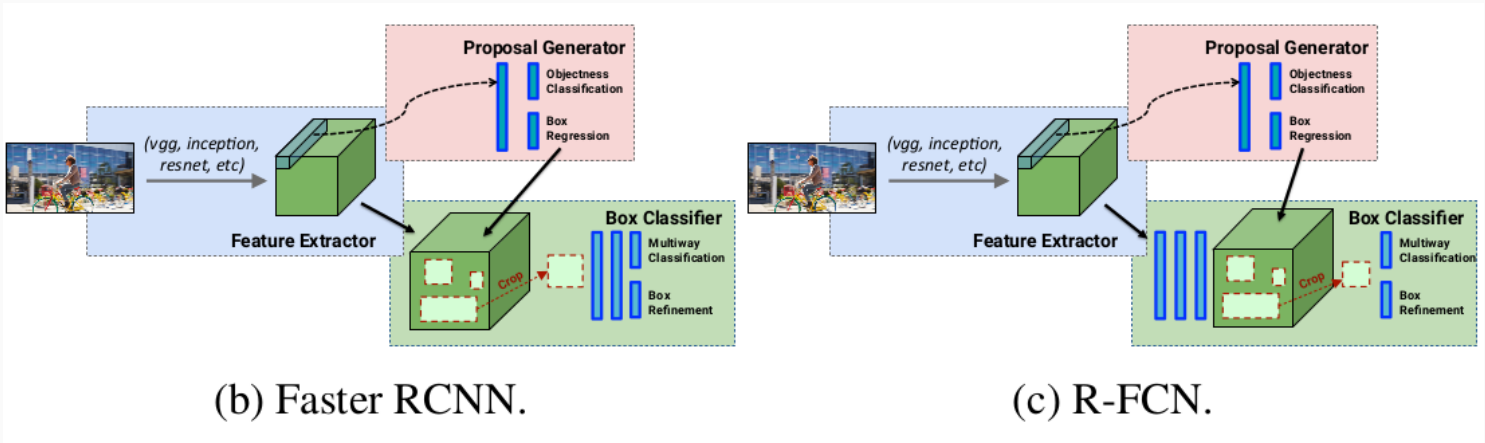


# СВЁРТОЧНЫЕ НЕЙРОННЫЕ СЕТИ

- ResNet:

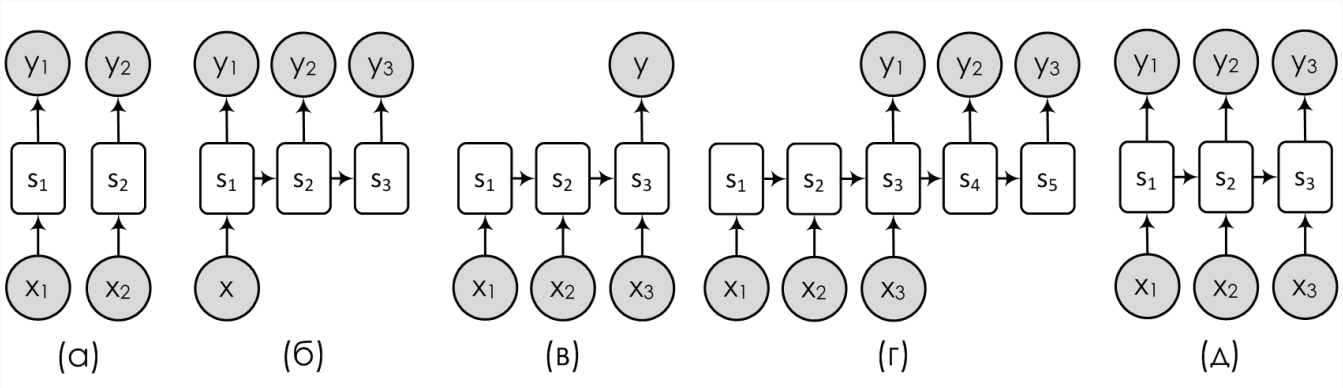



- Плюс другие целевые функции (например, сегментация):

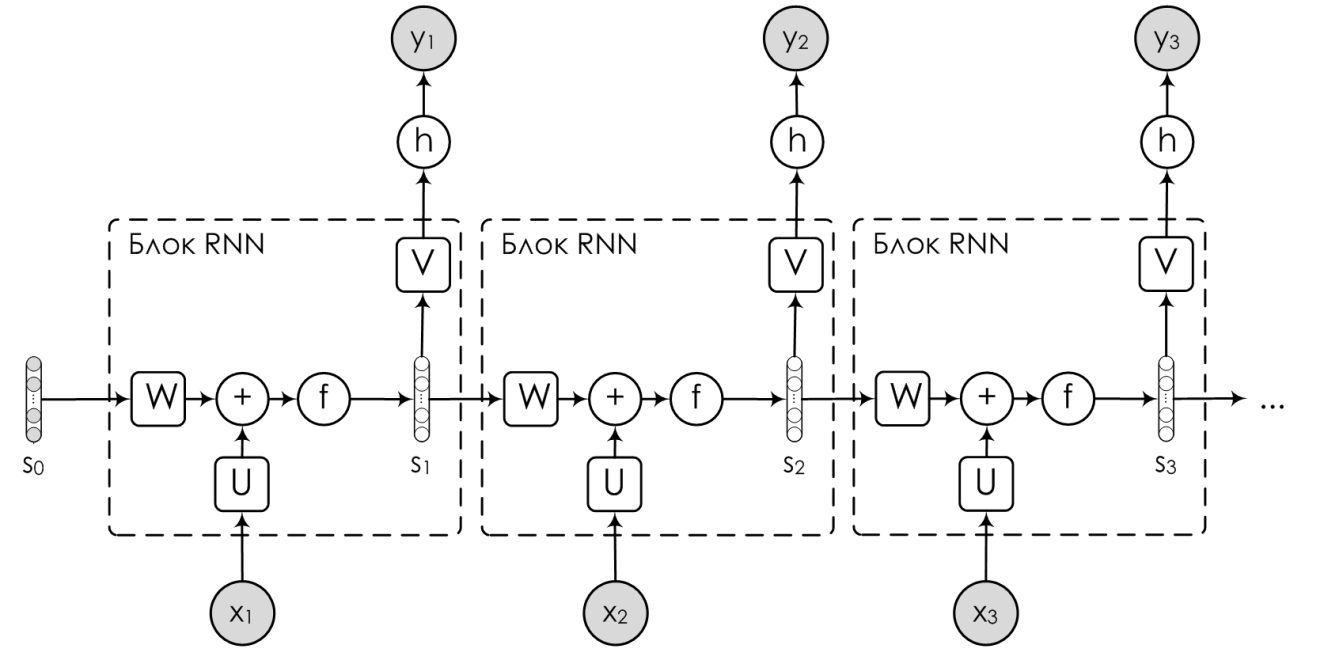


# РЕКУРРЕНТНЫЕ НЕЙРОННЫЕ СЕТИ

- Есть разные виды задач, основанных на последовательностях:

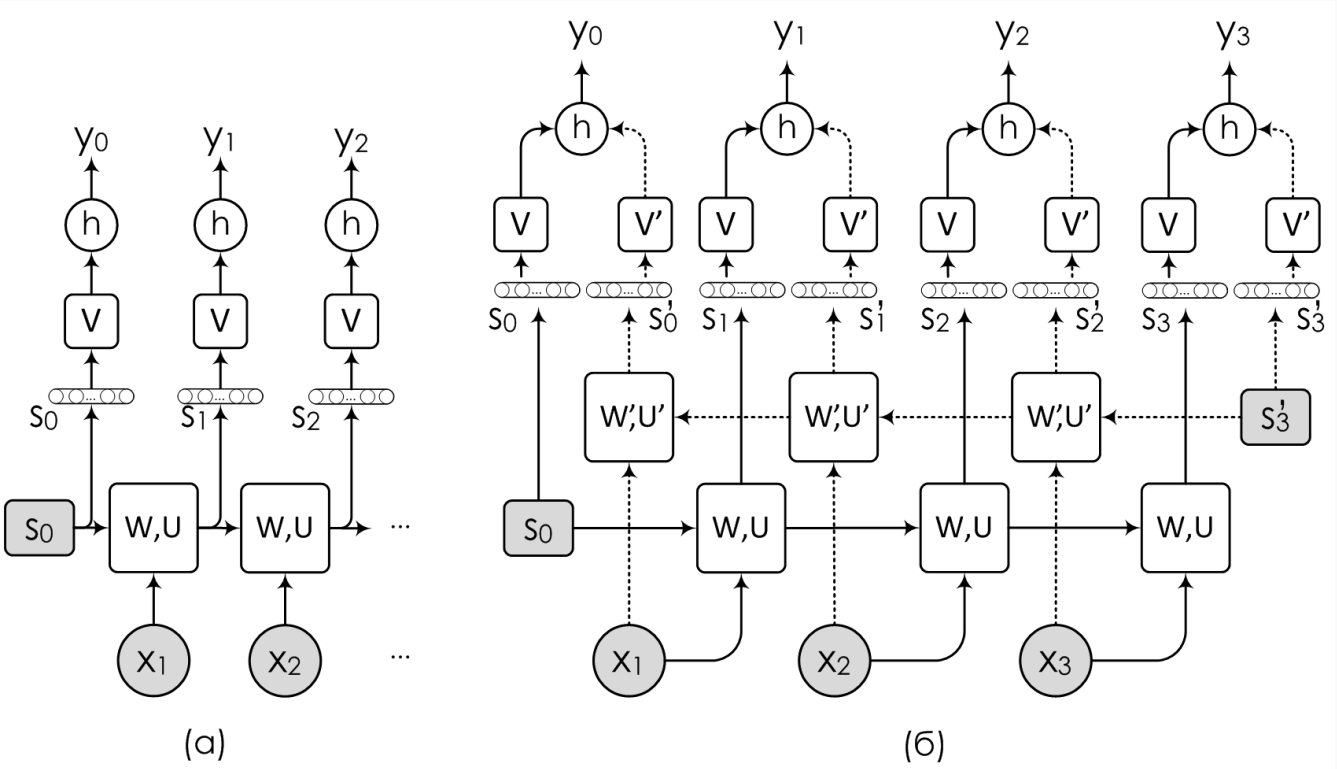



- “Простая” RNN:



# РЕКУРРЕНТНЫЕ НЕЙРОННЫЕ СЕТИ

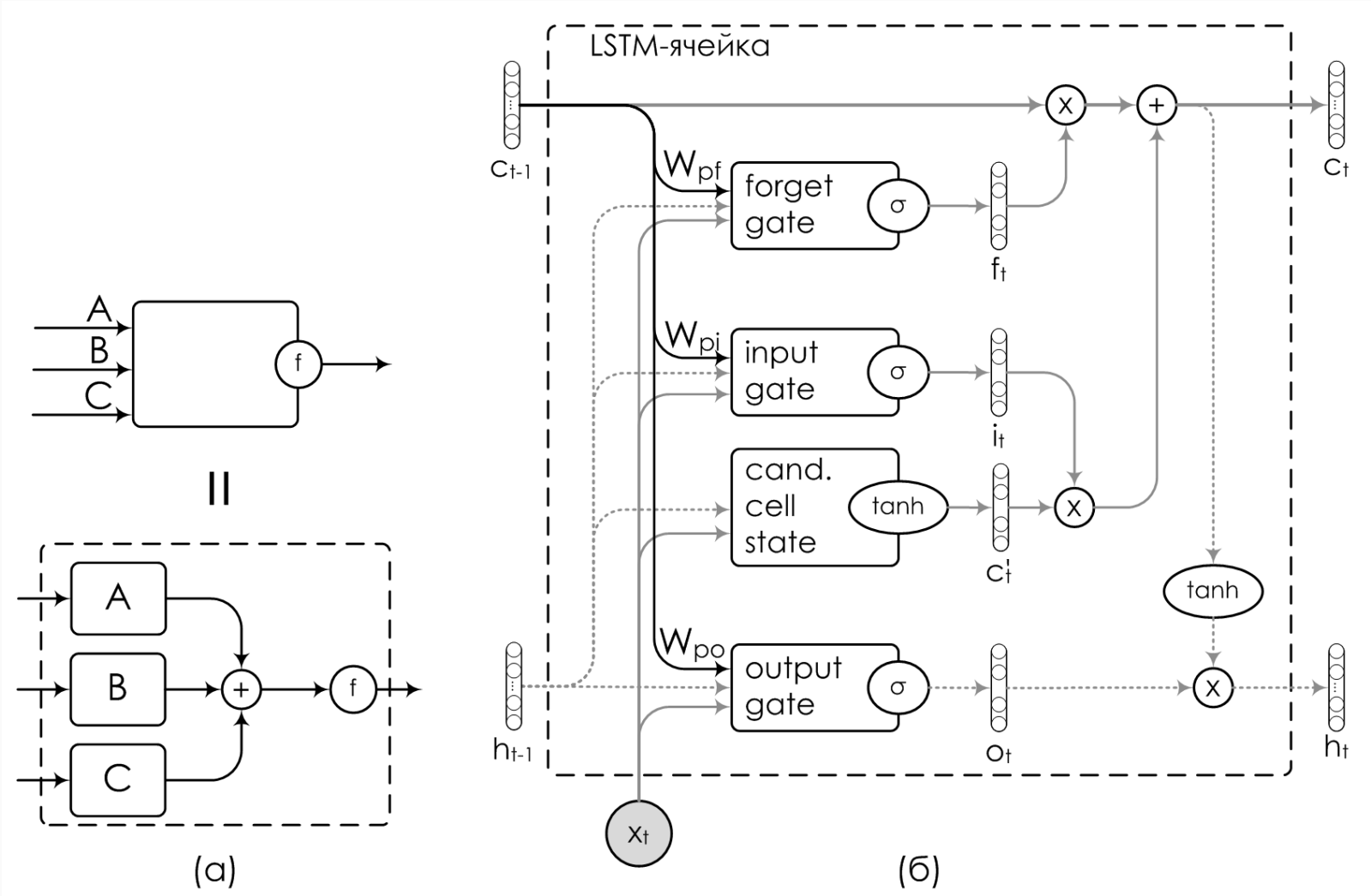
- Двунаправленная RNN:



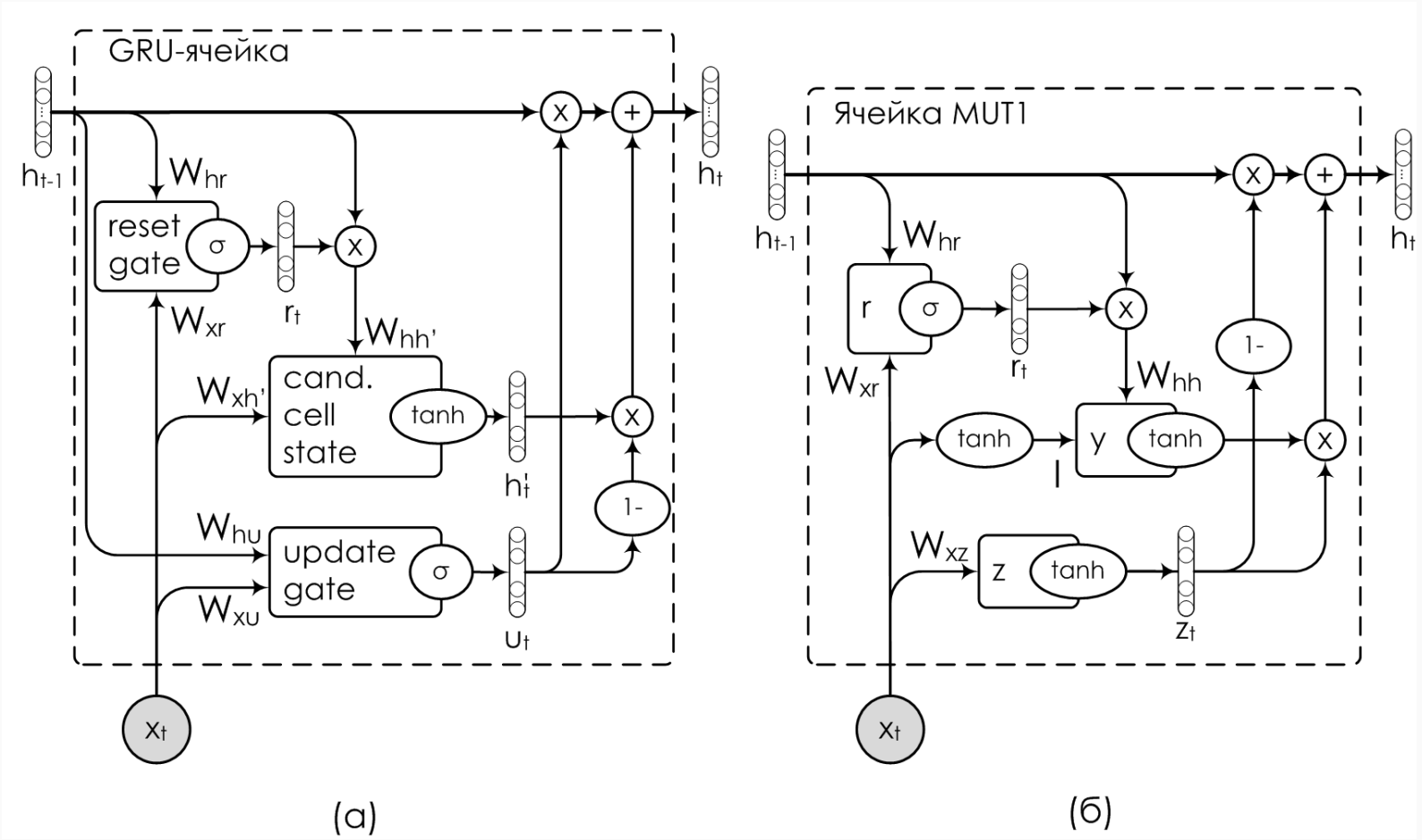

- Две проблемы RNN:
  - взрывающиеся градиенты (exploding gradients);
  - затухающие градиенты (vanishing gradients).

# РЕКУРРЕНТНЫЕ НЕЙРОННЫЕ СЕТИ

- Карусель константной ошибки – LSTM:

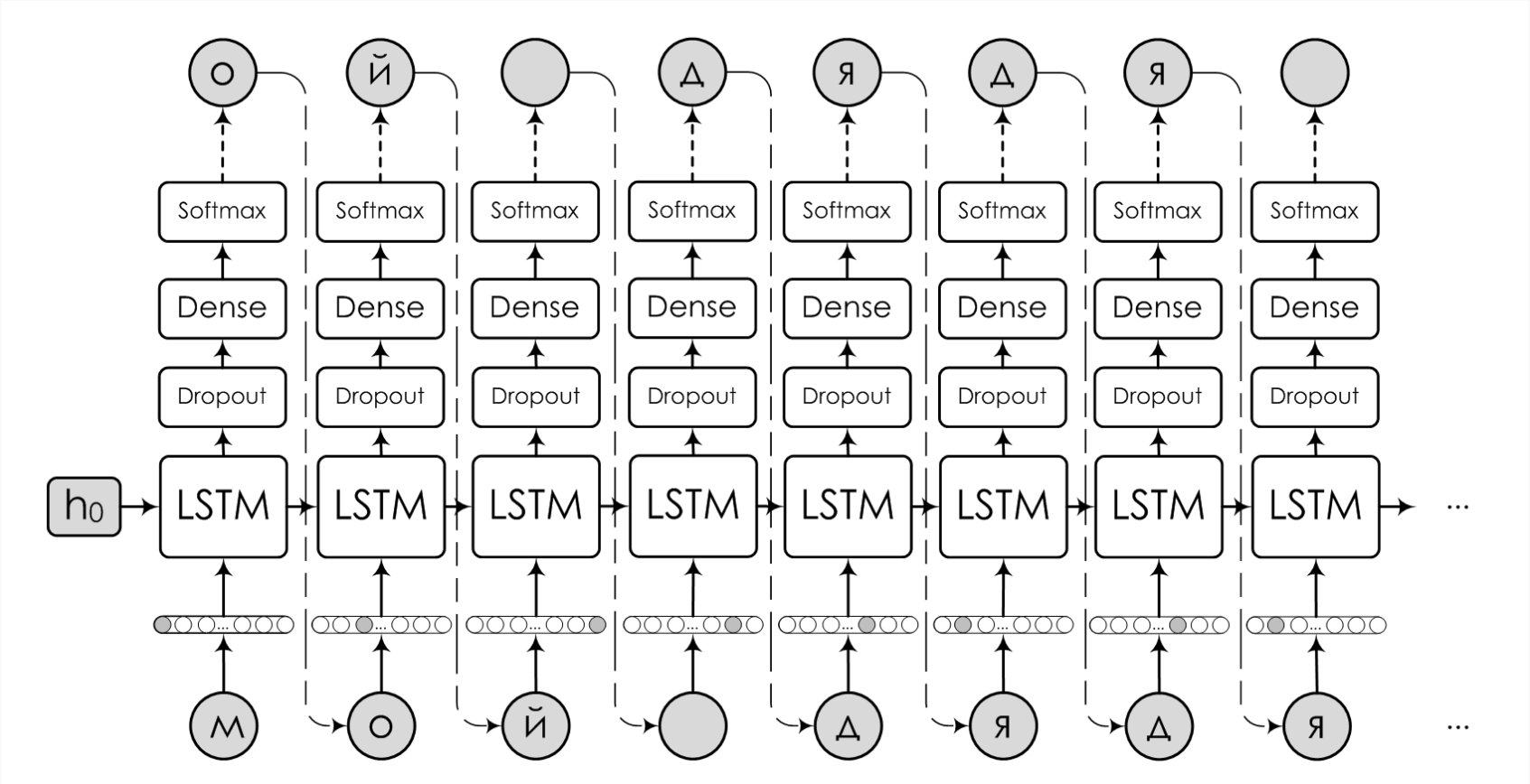


- Можно пытаться упростить – GRU:



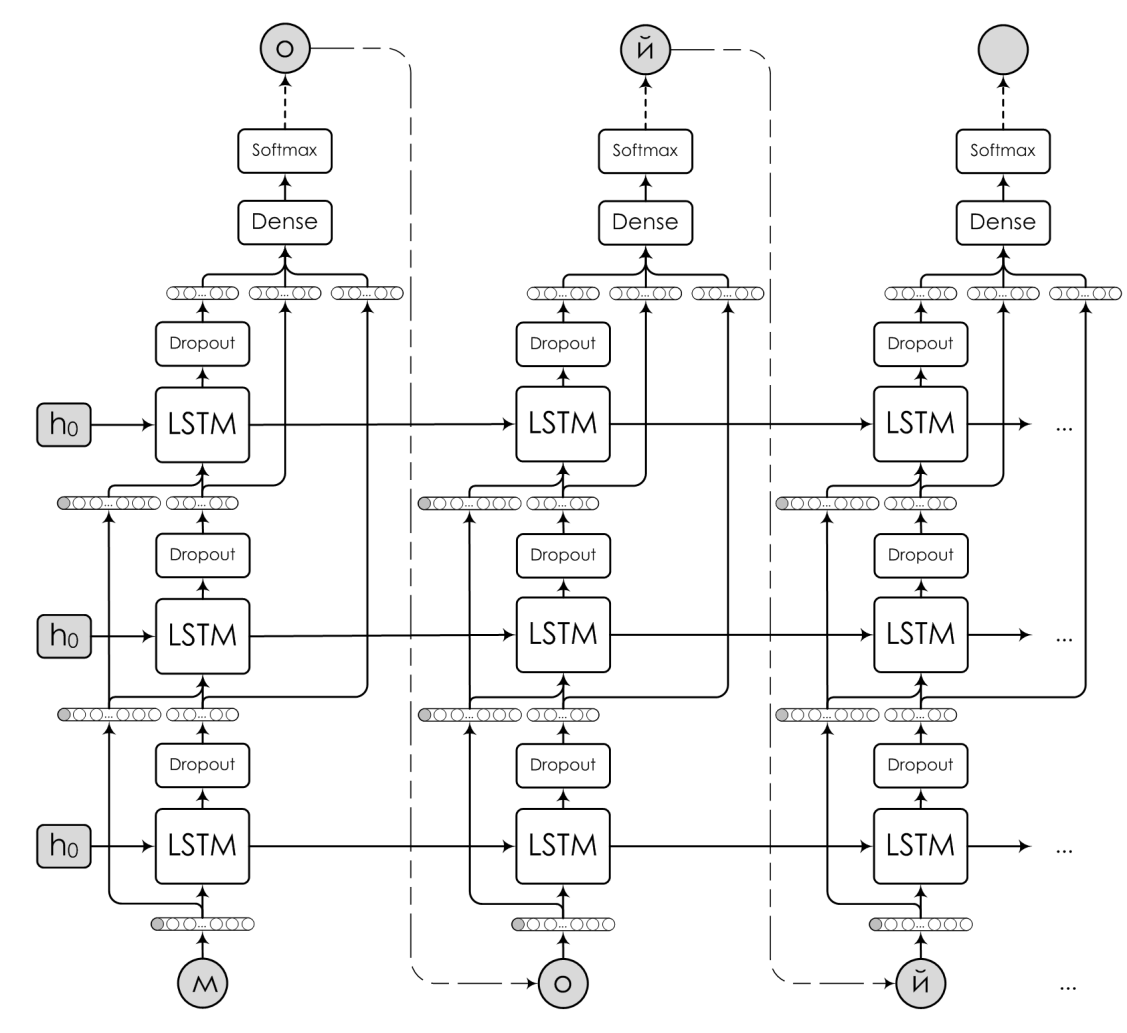
# СТРУКТУРА РЕКУРРЕНТНОЙ СЕТИ

- Простая seq2seq архитектура (Sutskever et al. 2014):



# СТРУКТУРА РЕКУРРЕНТНОЙ СЕТИ

- Менее простая:



# ОБРАБОТКА ЕСТЕСТВЕННОГО ЯЗЫКА

# ПЛАН

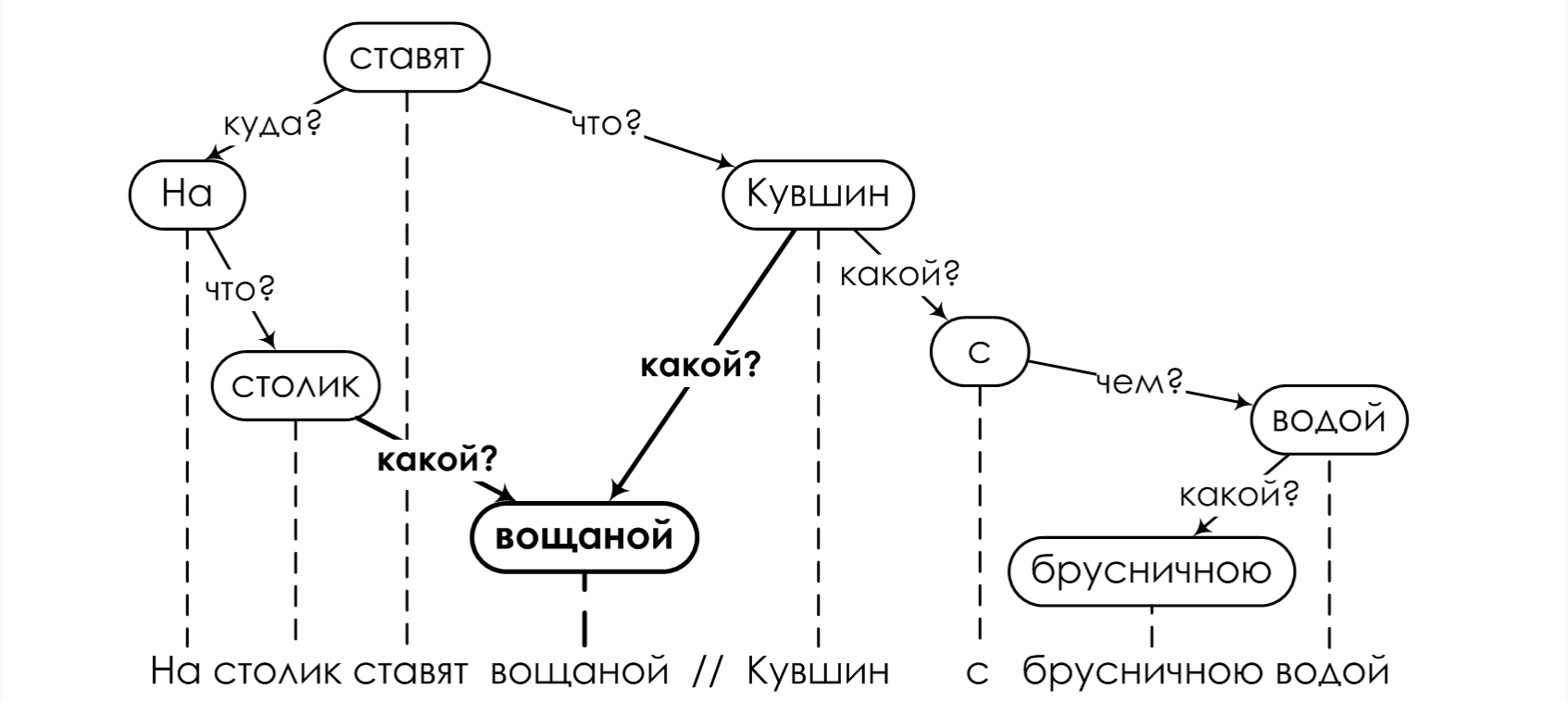
- Революция глубокого обучения, конечно же, не осталась в стороне от NLP.
- DL в NLP началось со стандартных архитектур (RNN, CNN), но затем часто мотивировало новые архитектуры.
- Их мы и постараемся сегодня в основном обсуждать.
- План:
  (1) задачи NLP;
  (2) распределённые представления слов;
  (3) распределённые представления коротких текстов и модели, основанные на символах;
  (4) современный NLP и deep learning: новые архитектуры.

# ЗАДАЧИ NLP

# ЗАДАЧИ NLP

- Синтаксические, более-менее хорошо определённые задачи:
  - *частеречная разметка* (part-of-speech tagging);
  - *морфологическая сегментация* (morphological segmentation);
  - *стемминг* (stemming) или *лемматизация* (lemmatization);
  - *выделение границ предложения* (sentence boundary disambiguation);
  - *пословная сегментация* (word segmentation);
  - *распознавание именованных сущностей* (named entity recognition);
  - *разрешение смысла слов* (word sense disambiguation);
  - *синтаксический парсинг* (syntactic parsing);
  - *разрешение кореференций* (coreference resolution).

- Но и для этого нужно понимание текста:




- Разрешение анафоры:
  - «мама вымыла раму, и теперь она блестит»;
  - «мама вымыла раму, и теперь она устала».

# Задачи NLP

- Более сложные задачи, которые ещё чаще требуют понимания, но правильные ответы и метрики качества легко придумать:
  - *языковые модели* (language models);
  - *анализ тональности* (sentiment analysis);
  - *выделение отношений* или *фактов* (relationship extraction, fact extraction);
  - *ответы на вопросы* (question answering).

# ЗАДАЧИ NLP

- Задачи, где нужно не только понять текст, но и породить новый текст:
  - собственно *порождение текста* (text generation);
  - *автоматическое реферирование* (automatic summarization);
  - *машинный перевод* (machine translation);
  - *диалоговые модели* (dialog and conversational models).
- И для всего этого многообразия и великолепия есть модели в виде глубоких нейронных сетей.

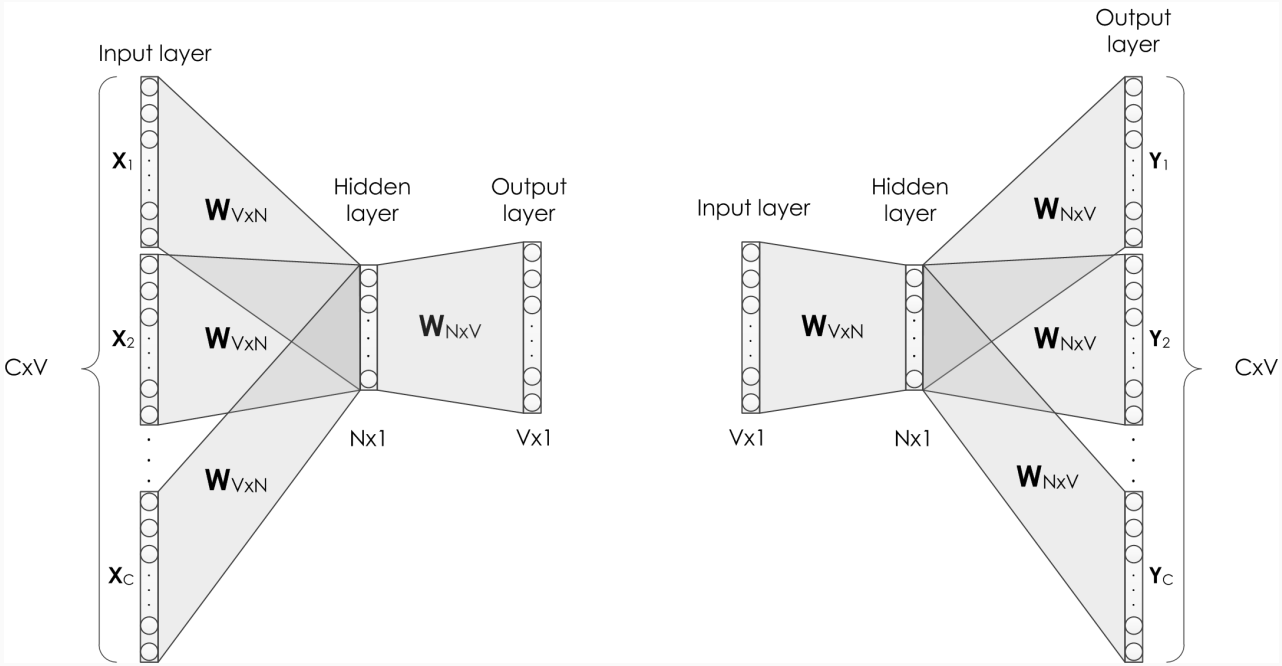
# Распределённые представления слов и других объектов

# WORD EMBEDDINGS

- Distributional hypothesis в лингвистике: слова с похожими значениями будут встречаться в похожих контекстах.
- *Распределённые представления слов* (distributed word representations, word embeddings) отображают слова в евклидово пространство $\mathbb{R}^d$, обычно $d$ порядка нескольких сотен:
  - началось в (Bengio et al. 2003; 2006), хотя подобные идеи были и раньше;
  - *word2vec* (Mikolov et al. 2013): обучаем веса, которые лучше всего решают простую задачу предсказания слова по его контексту: continuous bag-of-words (CBOW) и skip-gram;
  - *Glove* (Pennington et al. 2014): обучаем веса слов, раскладывая матрицу совместной встречаемости.

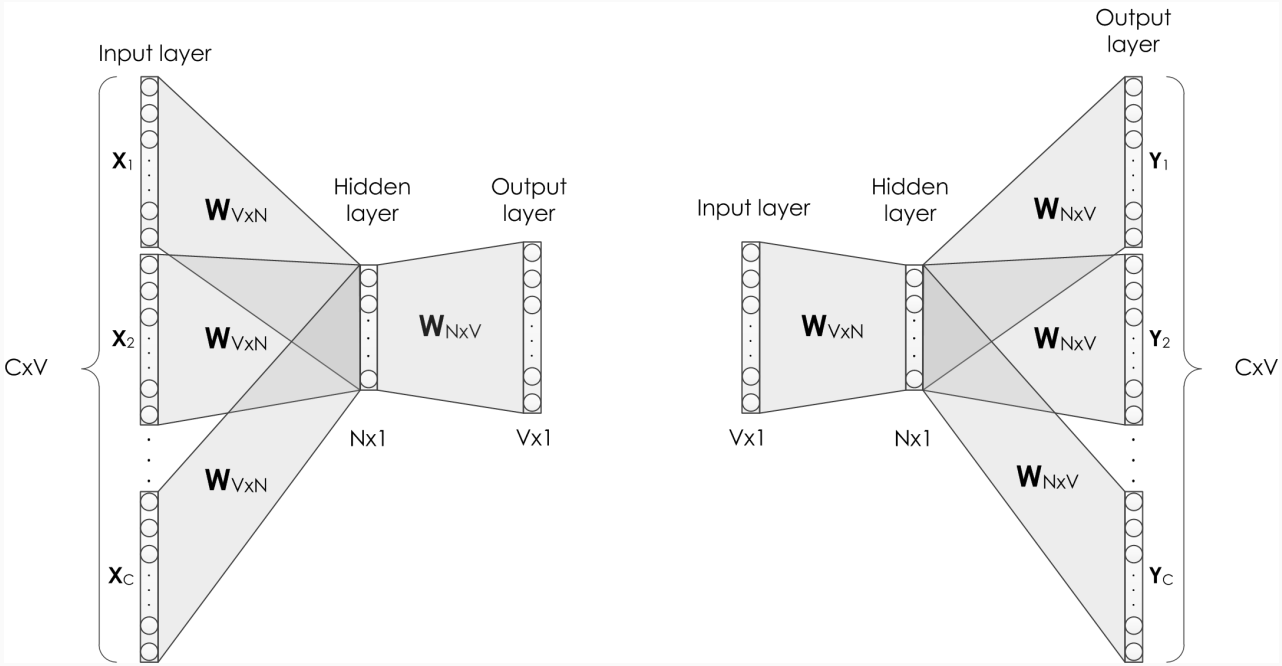
- CBOW предсказывает слово из локального контекста:
  - входы – one-hot представления слов размерности $V$;
  - скрытый слой – матрица представлений слов $W$;
  - выход скрытого слоя – среднее векторов слов контекста;
  - на выходе получаем оценку $u_j$ для каждого слова и берём $\mathrm{softmax}$:

$$\hat{p}(i|c_1, \dots, c_n) = \frac{\exp(u_j)}{\sum_{j'=1}^{V} \exp(u_{j'})}.$$



- Skip-gram предсказывает слова контекста из текущего слова:
  - предсказываем каждое слово контекста из центрального;
  - теперь несколько мультиномиальных распределений и по $\mathrm{softmax}$ для каждого слова контекста:

$$\hat{p}(c_k|i) = \frac{\exp(u_{kc_k})}{\sum_{j'=1}^{V} \exp(u_{j'})}.$$



# ПРЕДСТАВЛЕНИЯ СЛОВ

- Как обучить такую модель?
- Например, в skip-gram выбираем $\theta$ так, чтобы максимизировать

$$L(\theta) = \prod_{i \in D} \left( \prod_{c \in C(i)} p(c \mid i; \theta) \right) = \prod_{(i,c) \in D} p(c \mid i; \theta),$$

параметризуем

$$p(c \mid i; \theta) = \frac{\exp(\tilde{\mathbf{w}}_c^\top \mathbf{w}_i)}{\sum_{c'} \exp(\tilde{\mathbf{w}}_{c'}^\top \mathbf{w}_i)}.$$

# ПРЕДСТАВЛЕНИЯ СЛОВ

- Теперь общее правдоподобие равно

$$\arg\max_\theta \prod_{(i,c)\in D} p(c \mid i;\theta) = \arg\max_\theta \sum_{(i,c)\in D} p(c \mid i;\theta) =$$

$$= \arg\max_\theta \sum_{(i,c)\in D} \left( \exp(\tilde{\mathbf{w}}_c^\top \mathbf{w}_i) - \log \sum_{c'} \exp(\tilde{\mathbf{w}}_{c'}^\top \mathbf{w}_i) \right),$$

  и его можно максимизировать отрицательным сэмплированием (negative sampling).

- Вопрос: зачем нужно разделять векторы $\tilde{\mathbf{w}}$ и $\mathbf{w}$?

# ПРЕДСТАВЛЕНИЯ СЛОВ

- GloVe – пытаемся приблизить матрицу встречаемости $X \in \mathbb{R}^{V \times V}$:

$$p_{ij} = p(j \mid i) = \frac{X_{ij}}{X_i} = \frac{X_{ij}}{\sum_k X_{ik}}.$$

- Точнее, их отношения $\frac{p_{ij}}{p_{kj}}$.
- Пример на русской википедии:

| Слово $k$ | Число вхождений | | | Вероятности $p(k \mid ...)$, $\times 10^{-4}$ | | Отношение $\frac{p(k\mid\text{клуб})}{p(k\mid\text{команда})}$ |
|---|---|---|---|---|---|---|
| | Всего | Вместе с: | | | | |
| | | клуб | команда | клуб | команда | |
| футбол | 29988 | 54 | 34 | 18.0 | 11.3 | 1.588 |
| хоккей | 10957 | 16 | 7 | 6.39 | 14.6 | 2.286 |
| гольф | 2721 | 11 | 1 | 40.4 | 3.68 | 11.0 |
| корабль | 100127 | 0 | 30 | 0.0 | 3.00 | 0.0 |

# ПРЕДСТАВЛЕНИЯ СЛОВ

- Обучаем функцию $F(\mathbf{w}_i, \mathbf{w}_j; \tilde{\mathbf{w}}_k) = \frac{p_{ij}}{p_{kj}}$.
- Ещё проще – обучаем

$$F((\mathbf{w}_i - \mathbf{w}_j)^\top \tilde{\mathbf{w}}_k) = \frac{F\left(\mathbf{w}_i^\top \tilde{\mathbf{w}}_k\right)}{F\left(\mathbf{w}_j^\top \tilde{\mathbf{w}}_k\right)} = \frac{p_{ij}}{p_{kj}}.$$

- Это фактически должна быть экспонента:

$$\mathbf{w}_i^\top \tilde{\mathbf{w}}_k = \log(p_{ik}) = \log(X_{ik}) - \log(X_i).$$

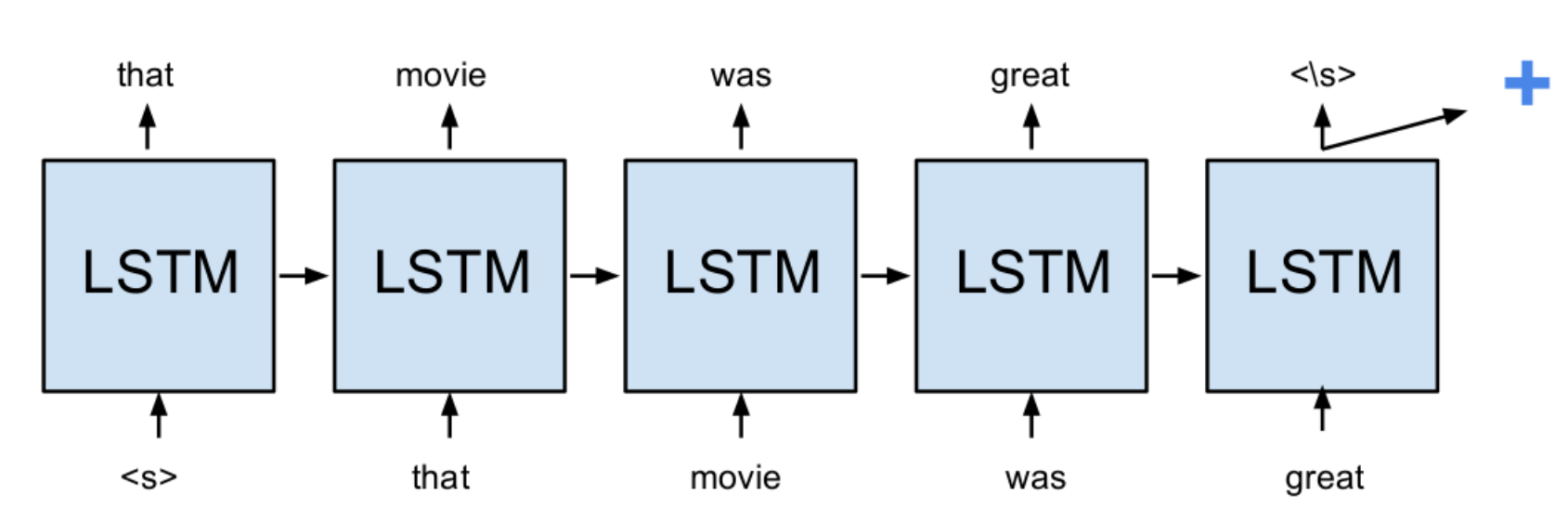
- Можно спрятать $\log(X_i)$ как $\mathbf{w}_i^\top \tilde{\mathbf{w}}_k + b_i + \tilde{b}_k = \log(X_{ik})$.
- И целевая функция у GloVe будет

$$J = \sum_{i,j=1}^{V} f(X_{ij}) \left(\mathbf{w}_i^\top \tilde{\mathbf{w}}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2.$$

- Демо: ближайшие соседи, геометрические соотношения.

# КАК ИСПОЛЬЗОВАТЬ ВЕКТОРЫ СЛОВ

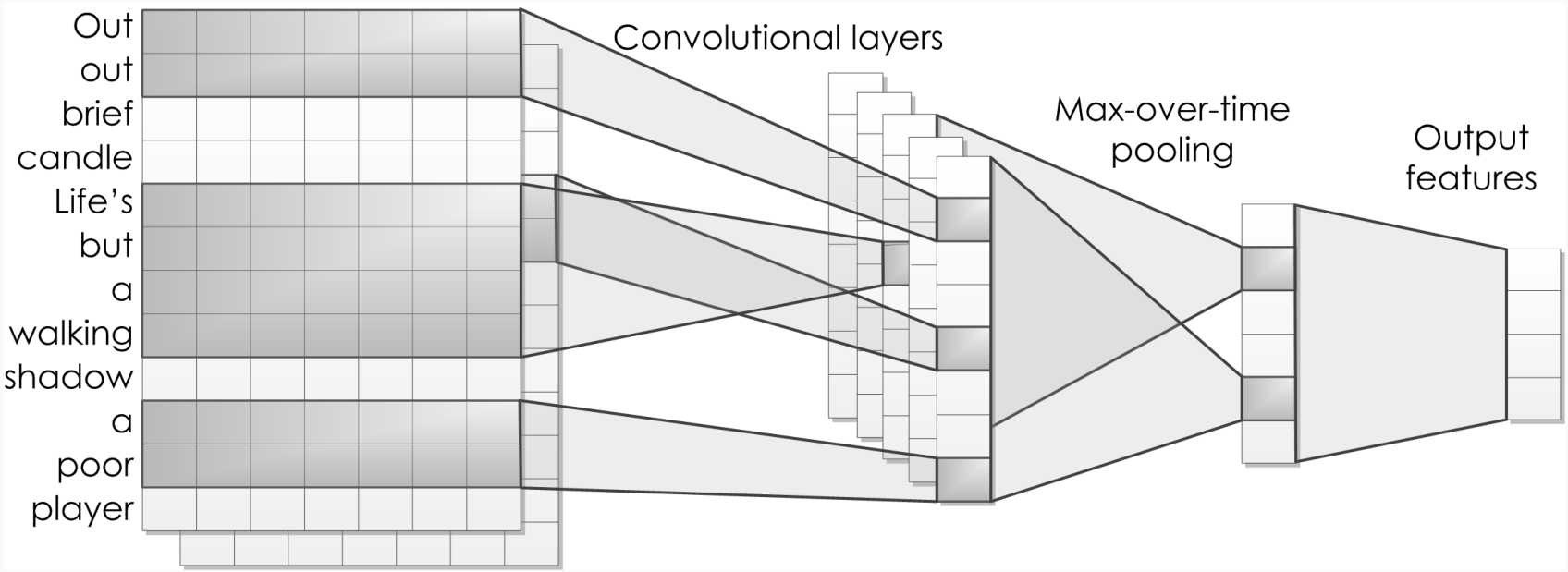
- Теперь можно просто строить нейронные сети поверх вложений.
- Например, стандартные LSTM для анализа тональности:




- Обучаем seq2seq для моделирования языка, затем используем последний выход или средний выход для тональности.

# КАК ИСПОЛЬЗОВАТЬ ВЕКТОРЫ СЛОВ

- Или CNN с одномерными свёртками:



# Вверх и вниз от представлений слов

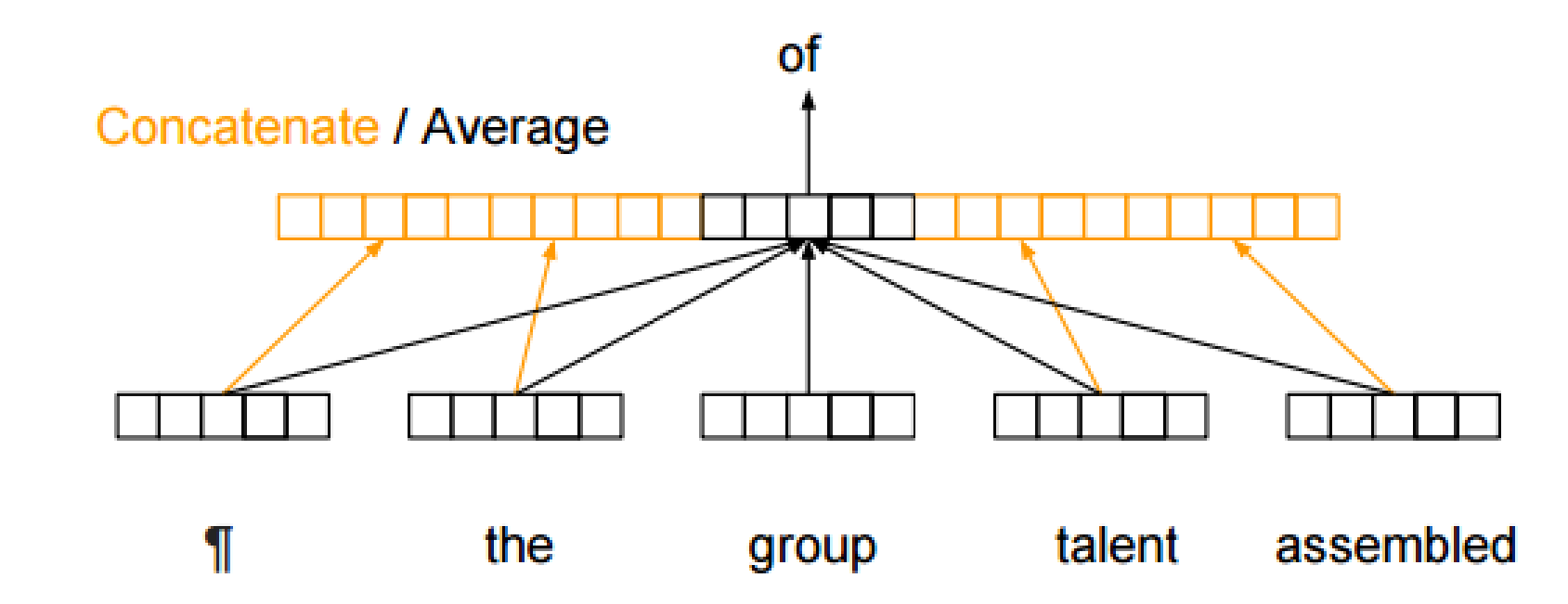
- Представления слов – первый шаг многих нейросетевых моделей в NLP.
- Но от представлений слов можно двигаться в обоих направлениях.
- Во-первых, предложение – не обязательно сумма своих слов.
- Во-вторых, слово не так уж атомарно, как модели хотелось бы думать.

# Представления предложений

- Как комбинировать векторы слов в векторы «кусков текста»?
- Простейшая идея: берём сумму и/или среднее представлений слов как представление предложения/абзаца:
  - это baseline в (Le and Mikolov 2014) и многих других работах;
  - разумный метод для коротких фраз в (Mikolov et al. 2013);
  - и довольно эффективно работает для реферирования в (Kageback et al. 2014).
- Совсем не так плохо, скорее всего как раз из-за геометрических соотношений.

# ПРЕДСТАВЛЕНИЯ ПРЕДЛОЖЕНИЙ

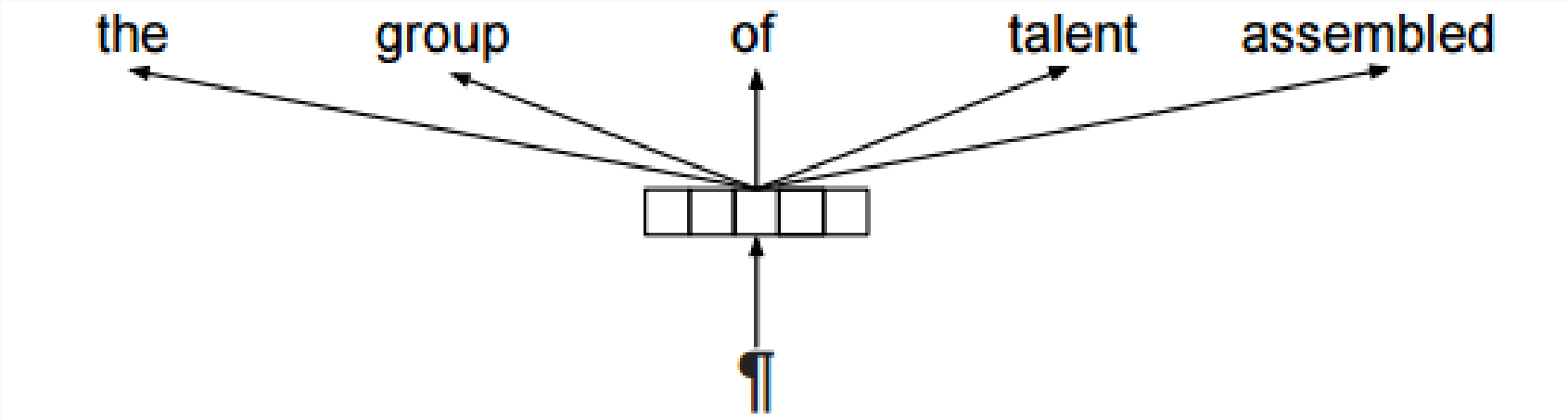
- Как комбинировать векторы слов в векторы «кусков текста»?
- Distributed Memory Model of Paragraph Vectors (PV-DM) (Le and Mikolov 2014):
  - вектор абзаца – дополнительный вектор для каждого абзаца;
  - служит «памятью» для более глобального контекста.



# ПРЕДСТАВЛЕНИЯ ПРЕДЛОЖЕНИЙ

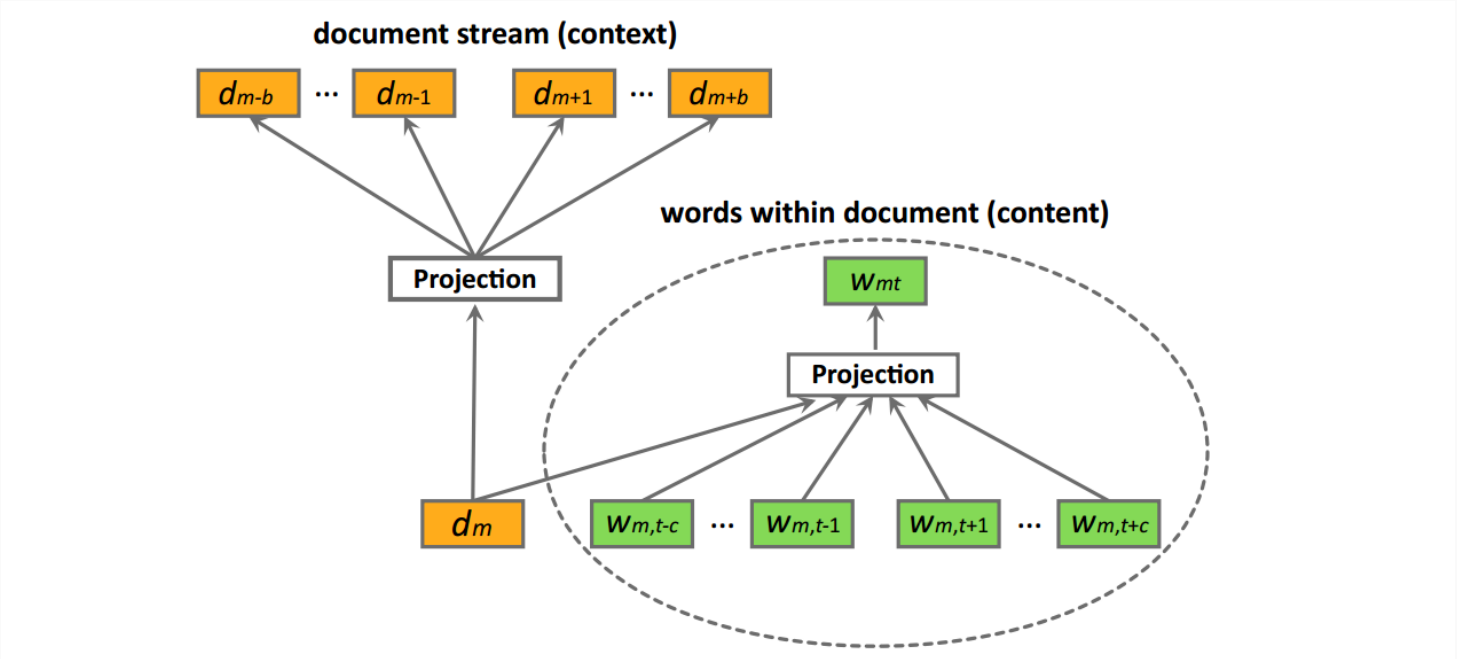
- Как комбинировать векторы слов в векторы «кусков текста»?

- Distributed Bag of Words Model of Paragraph Vectors (PV-DBOW) (Le and Mikolov 2014):
  - модель предсказывает слова, случайно выбранные из данного абзаца;
  - и вектор абзаца помогает предсказывать слова из этого абзаца во всех локальных контекстах.



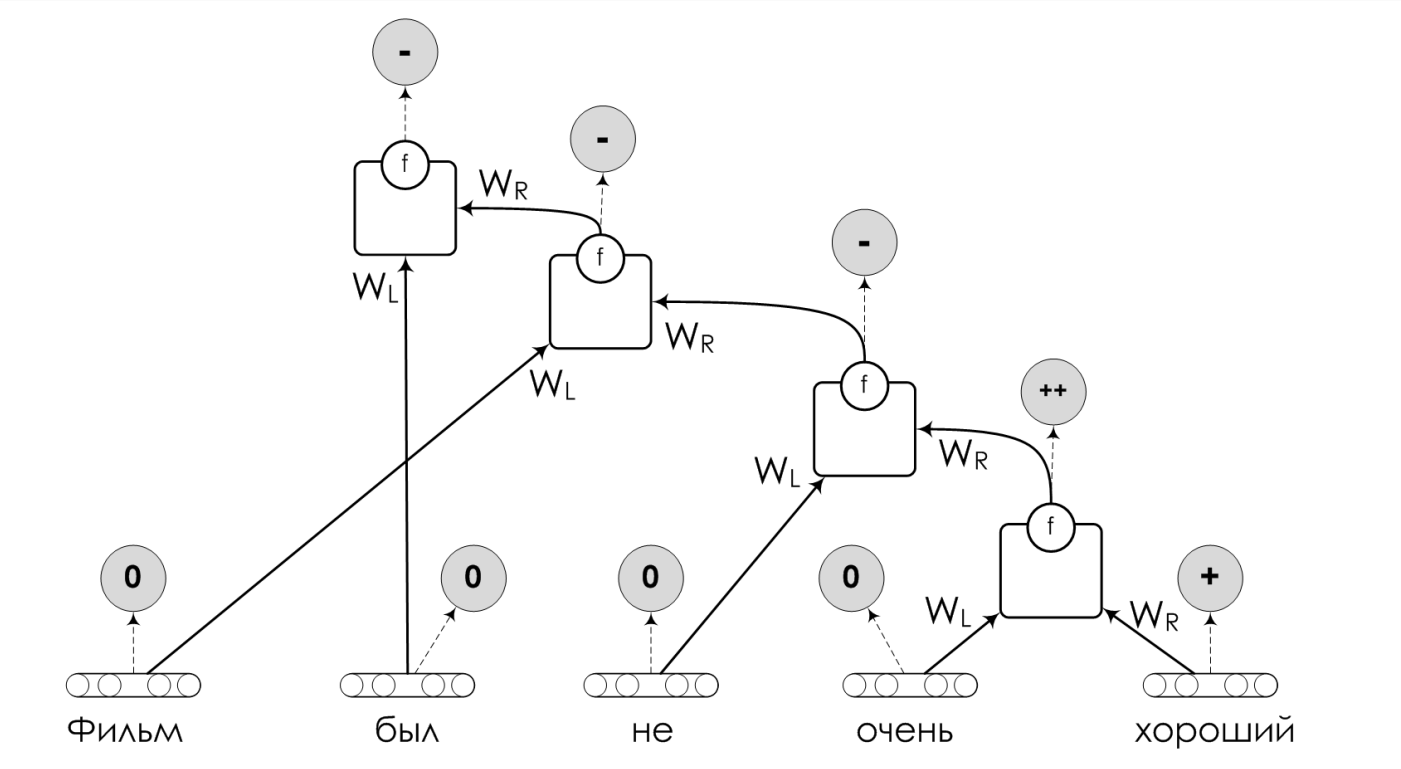
# ПРЕДСТАВЛЕНИЯ ПРЕДЛОЖЕНИЙ

- Как комбинировать векторы слов в векторы «кусков текста»?
  - Свёрточные архитектуры (Ma et al., 2015; Kalchbrenner et al., 2014).
  - (Kiros et al. 2015): skip-thought векторы, обучаются из skip-gram векторов на предложениях.
  - (Djuric et al. 2015): моделирует большие потоки текстов иерархическими нейросетевыми моделями.



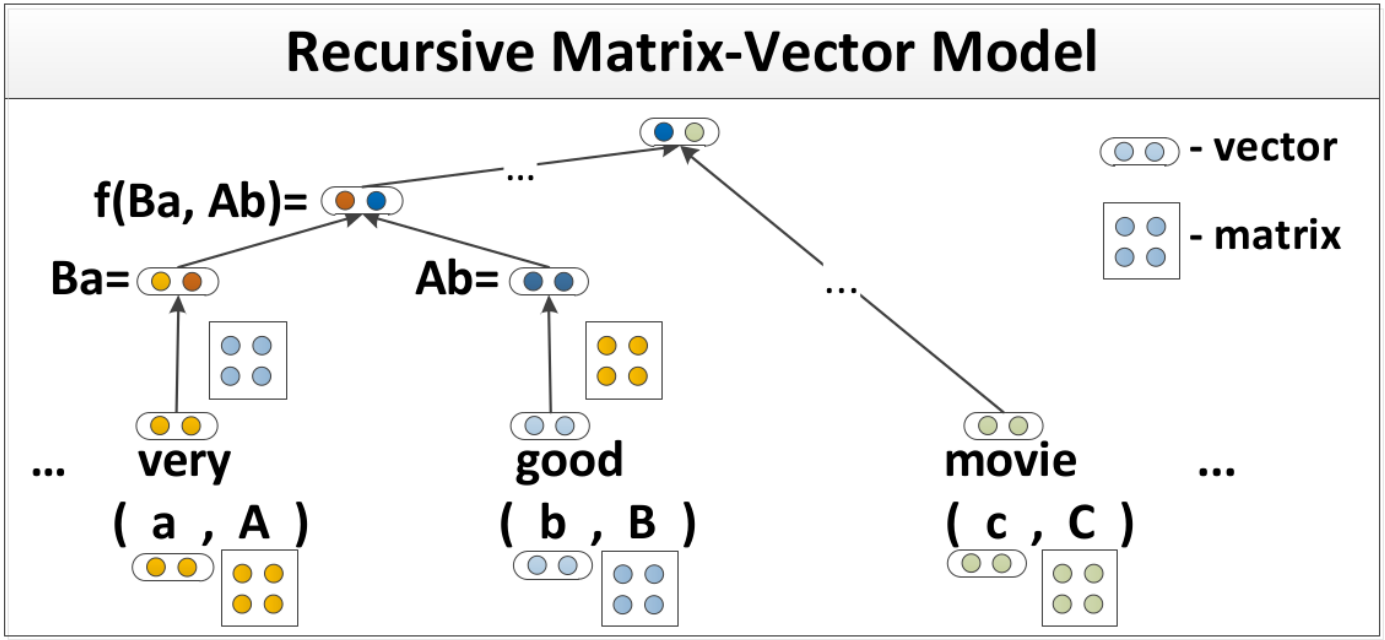
# ГЛУБОКИЕ РЕКУРСИВНЫЕ СЕТИ

- *Рекурсивные нейронные сети* (recursive neural networks; Socher et al., 2012):
  - нейронная сеть сворачивает представление куска текста в двух поддеревьях, проходя от слов до корня дерева синтаксического разбора.



# ГЛУБОКИЕ РЕКУРСИВНЫЕ СЕТИ

- *Рекурсивные нейронные сети* (recursive neural networks; Socher et al., 2012):
  - можно представлять узел матрицей и вектором;
  - в целом очень эффективный подход к анализу тональности (Socher et al. 2013).



# ГЛУБОКИЕ РЕКУРСИВНЫЕ СЕТИ

- Глубокие рекурсивные сети для анализа тональности (Irsoy, Cardie, 2014).
- Первая идея: отделим листья от внутренних узлов; вместо применения одних и тех же весов

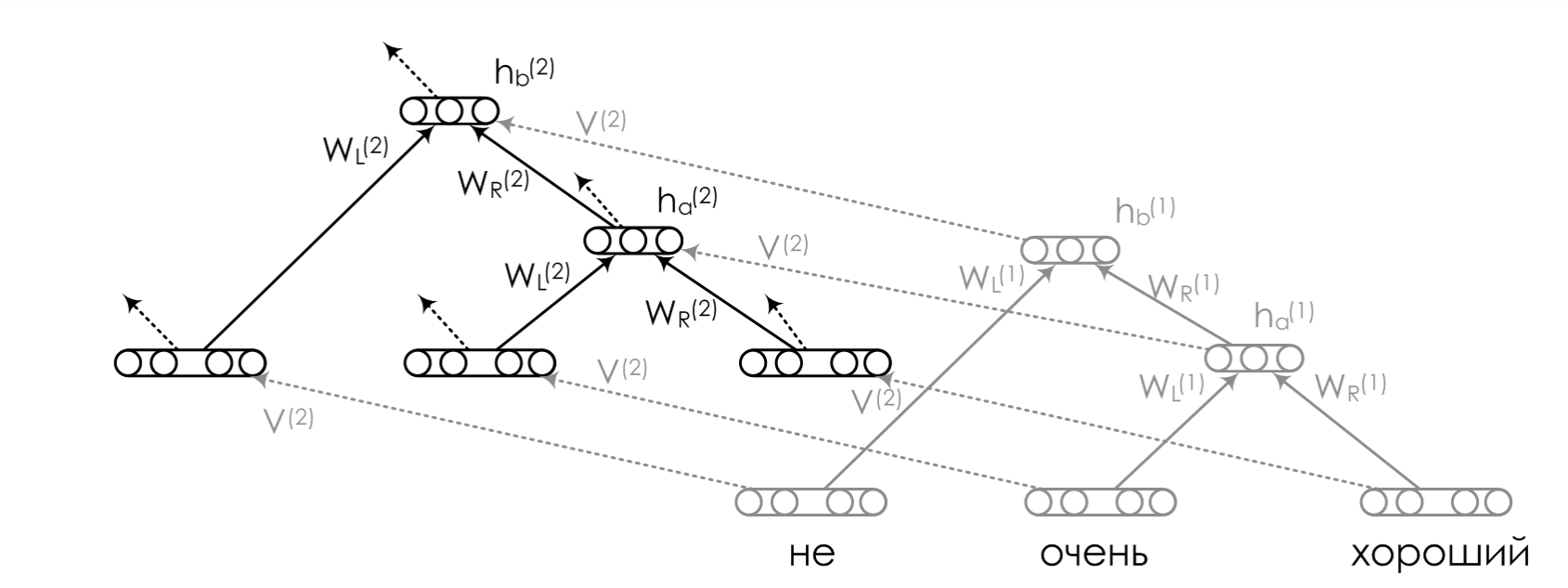
$$\mathbf{x}_v = f(W_L \mathbf{x}_{l(v)} + W_R \mathbf{x}_{r(v)} + \mathbf{b})$$

теперь будут разные матрицы для листьев (слов) и внутренних узлов:

  - теперь внутренняя размерность может быть меньше;
  - и можно использовать ReLU, т.к. теперь нет проблемы с разреженными входами и плотными внутренними узлами.

# ГЛУБОКИЕ РЕКУРСИВНЫЕ СЕТИ

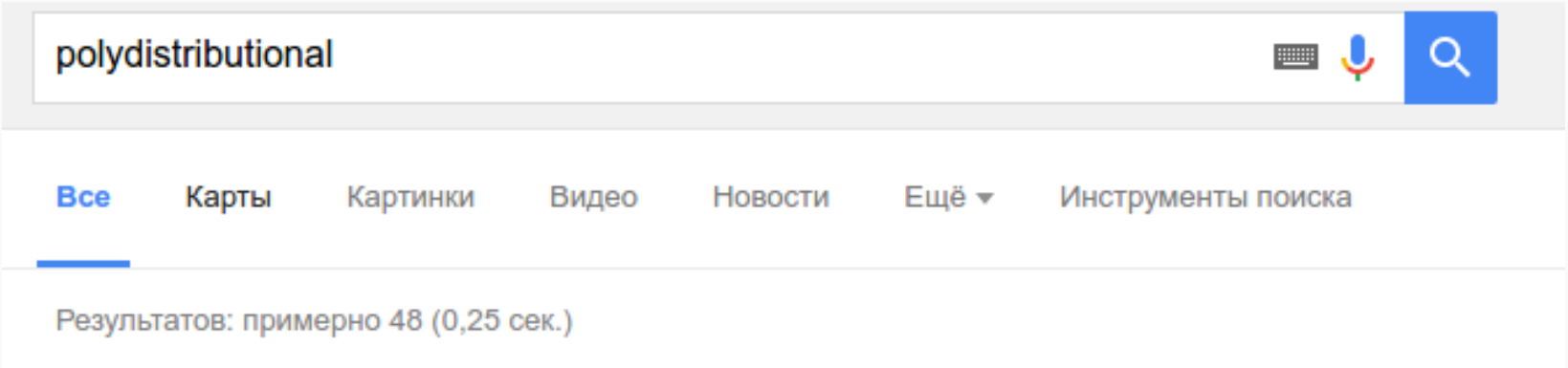
- Вторая идея – добавим в иерархическое представление глубины: $h_v^{(i)} = f(W_L^{(i)} h_{l(v)}^{(i)} + W_R^{(i)} h_{r(v)}^{(i)} + V^{(i)} h_v^{(i-1)} + \mathbf{b}^{(i)})$.




- Прекрасная архитектура для анализа тональности… если, конечно, есть деревья разбора.
- Stanford Sentiment TreeBank есть; а по-русски непонятно…

# CHARACTER-LEVEL MODELS

- Важные недостатки векторов слов:
  - векторы независимы, а слова нет (морфология);
  - то же относится к словам не из словаря (новым, очень редким);
  - модель очень большая получается, если все слова обучать.
- Например, слово “polydistributional” даёт 48 результатов в Google, так что вы его скорее всего никогда не видели, и обучаться не на чем:



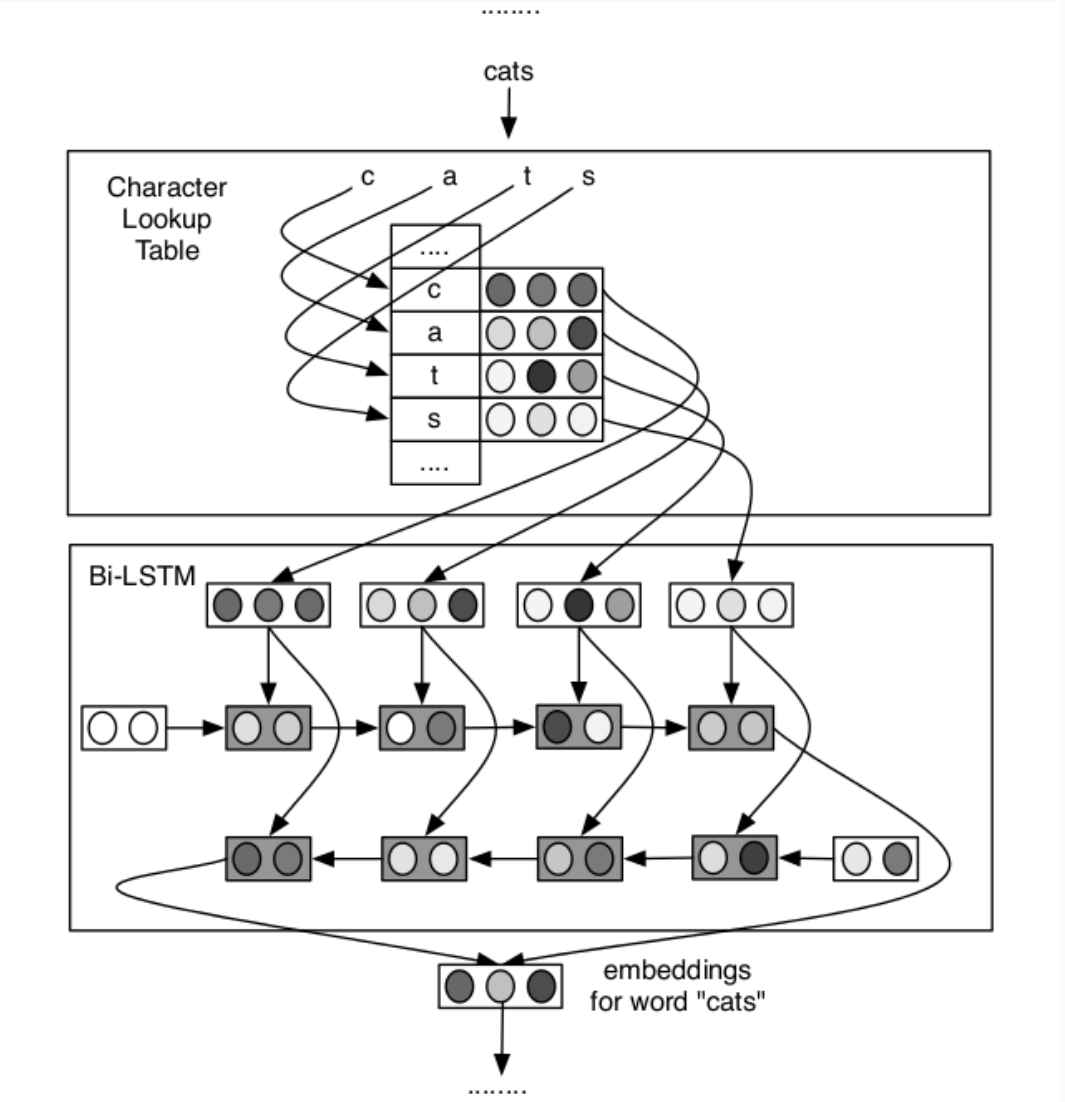

- Понимаете, что оно значит? Я тоже понимаю.

# CHARACTER-LEVEL MODELS

- Отсюда идея *character-level representations*:
  - сначала раскладывали слово на морфемы (Luong et al. 2013; Botha and Blunsom 2014; Soricut and Och 2015);
  - но тогда надо делать морфологический анализ, тоже непросто;
  - два естественных подхода на буквах: LSTM/GRU и CNN;
  - в любом случае, модель медленная, но к каждому слову применять не обязательно, можно частично закешировать заранее.

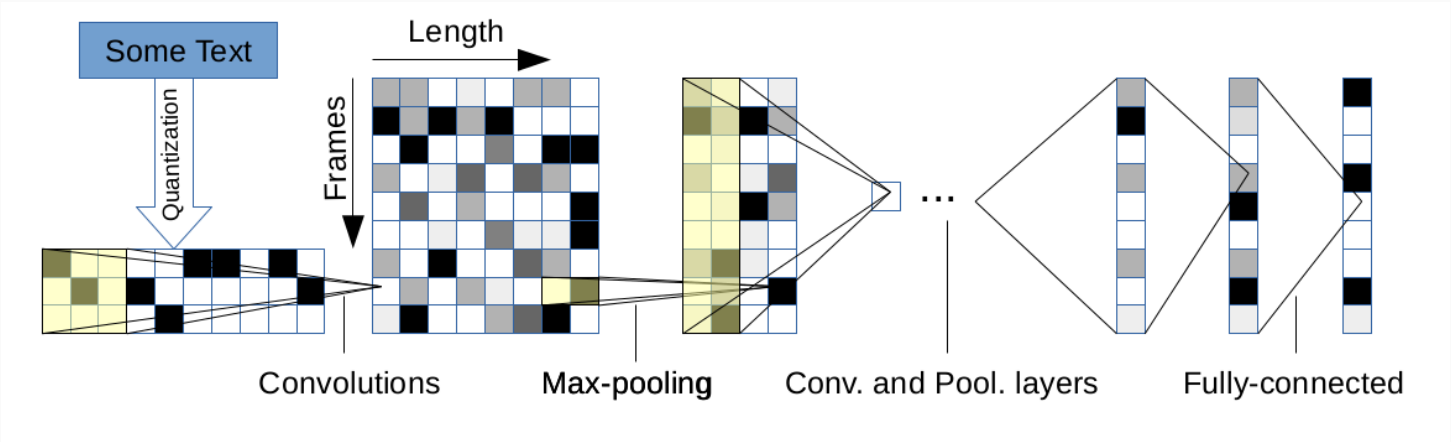
# CHARACTER-LEVEL MODELS

- C2W (Ling et al. 2015) основано на двунаправленных LSTM:



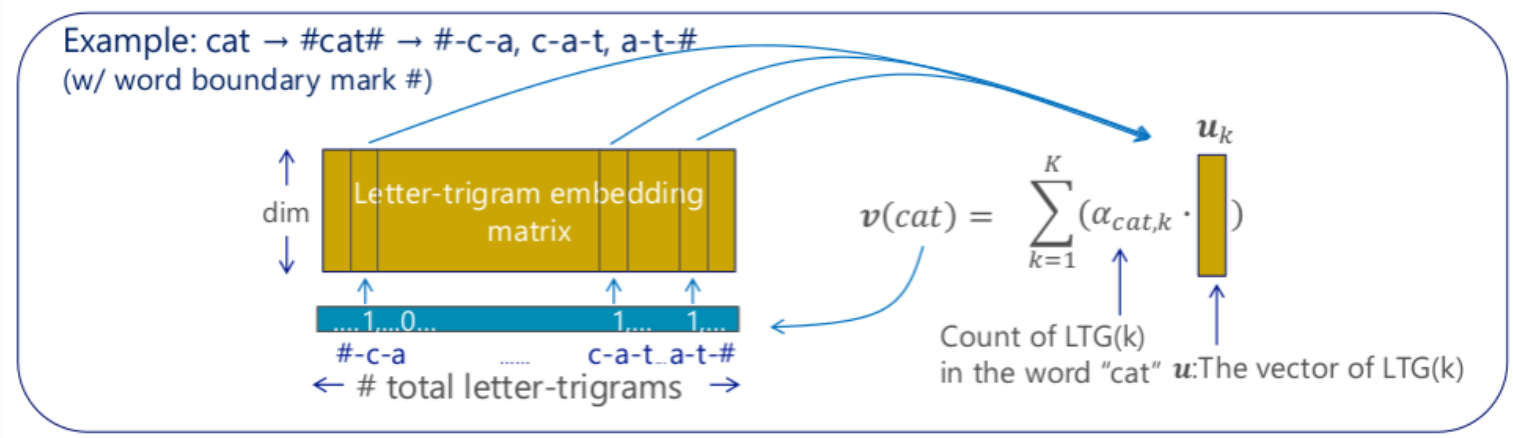
# CHARACTER-LEVEL MODELS

- ConvNet (Zhang et al. 2015): понимание текста с нуля, с букв, целиком на CNN.



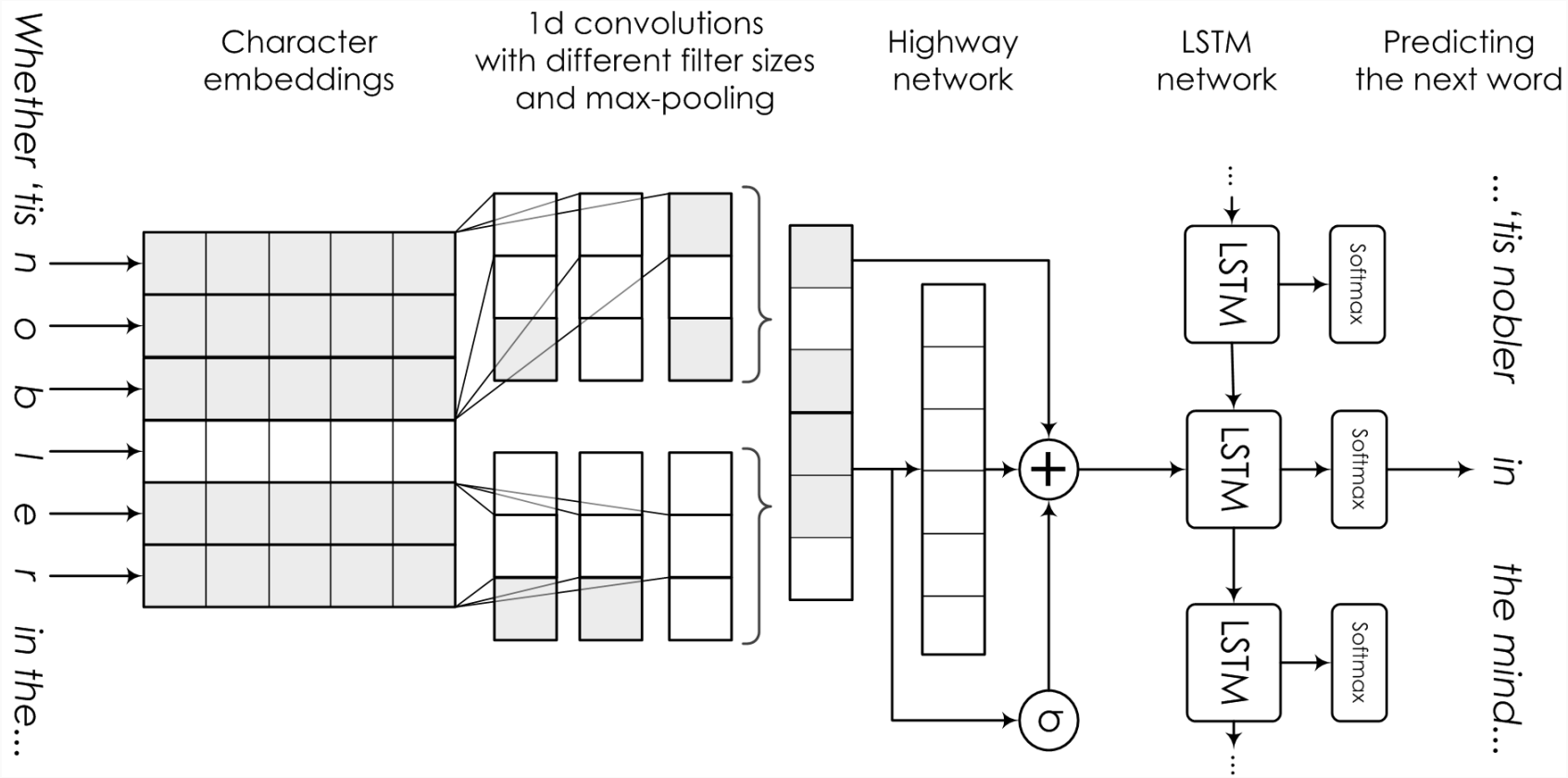

- Character-level модели становятся всё важнее и нужнее, особенно для (1) морфологически богатых языков и (2) порождённых пользователями текстов.

- Подход *Deep Structured Semantic Model* (DSSM) (Huang et al., 2013; Gao et al., 2014a; 2014b):
  - sub-word embeddings: представим слово как множество триграмм букв;
  - словарь теперь $|V|^3$ (десятки тысяч вместо миллионов), но коллизии очень редки;
  - представление устойчиво к опечаткам и т.п. (очень важно для порождённых пользователями текстов).



# ПРИМЕР МОДЕЛИ: KIM ET AL., 2015

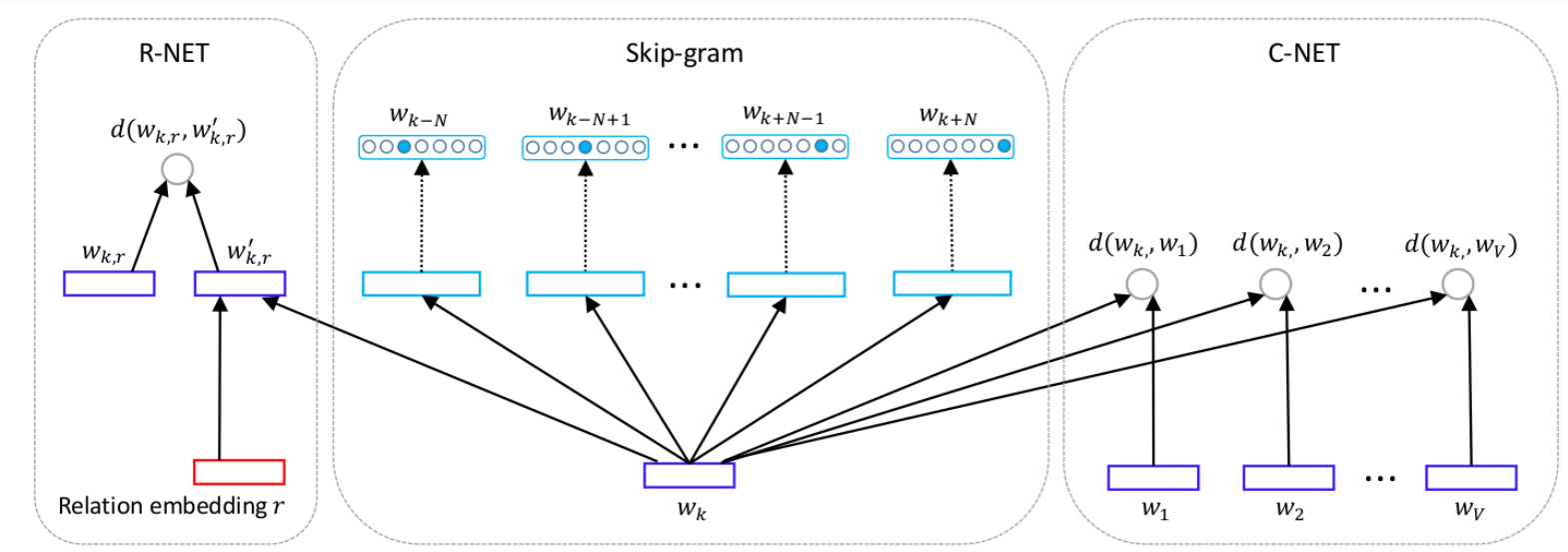
- Пример современной character-based языковой модели (Kim et al., 2015):




- Объединяет CNN, RNN, highway networks, embeddings...

# ВЕКТОРА СЛОВ С ВНЕШНЕЙ ИНФОРМАЦИЕЙ

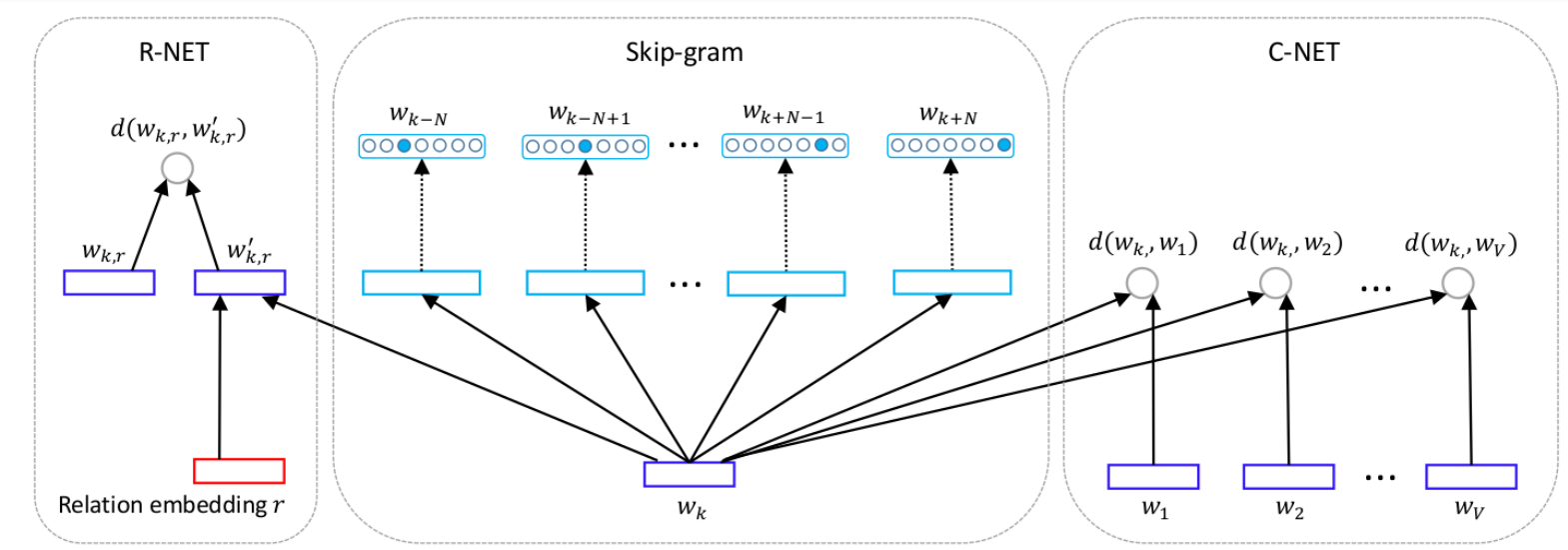
- Другие модификации представлений слов добавляют внешнюю информацию.
- Например, модель RC-NET (Xu et al. 2014) расширяет skip-gram семантическими и синтаксическими отношениями и знаниями.



# ВЕКТОРА СЛОВ С ВНЕШНЕЙ ИНФОРМАЦИЕЙ

- И базовый *word2vec* получает регуляризатор для каждого отношения в базе, пытаясь выразить его линейным соотношением:

$$\mathbf{w}_{\text{Hinton}} - \mathbf{w}_{\text{Wimbledon}} \approx r_{\text{born at}} \approx \mathbf{w}_{\text{Euler}} - \mathbf{w}_{\text{Basel}}$$



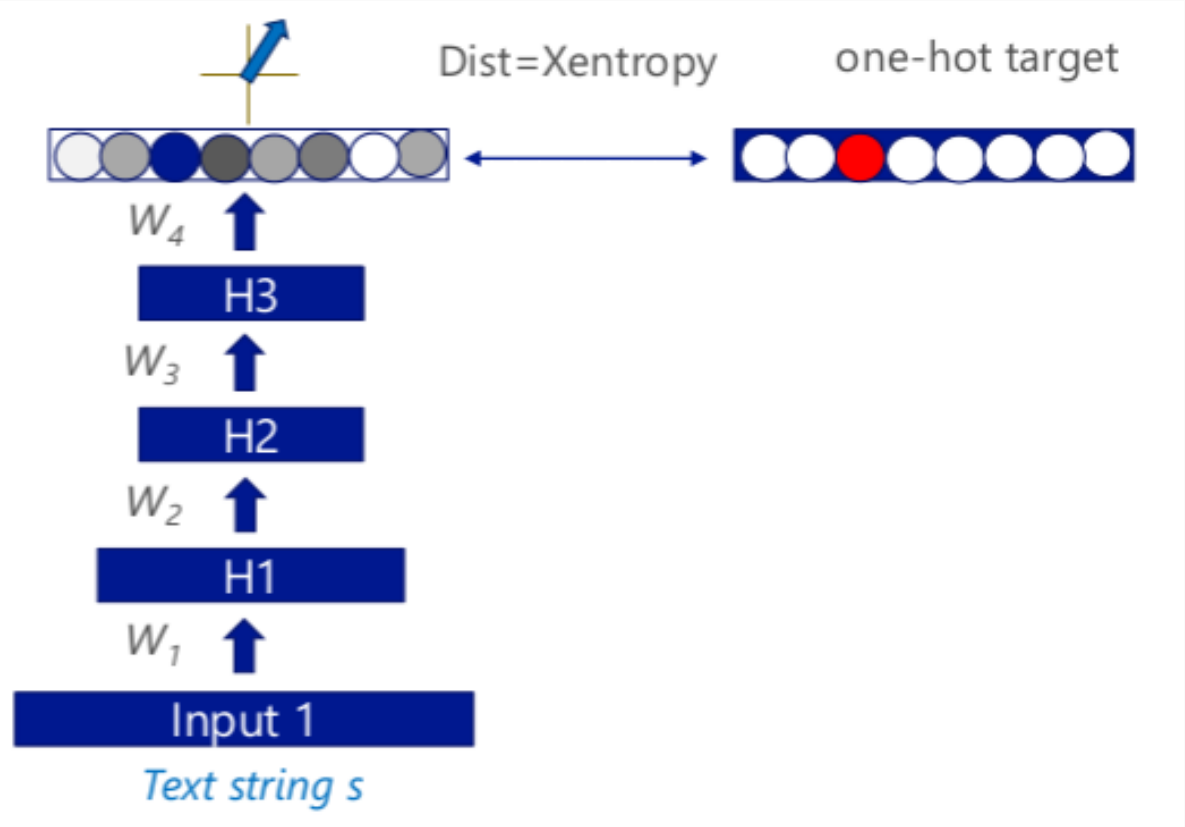
# РАЗРЕШЕНИЕ СМЫСЛА СЛОВ

- Другая важная проблема для всех моделей выше – омонимы.
- Как различить разные смыслы слова? Беда:
  - в модели обычно обучится только один смысл;
  - давайте проверим ближайших соседей слова **коса** и других омонимов.
- Надо добавить *скрытые* переменные для разных возможных смыслов и потом вывести их из контекста.
- Чтобы обучить смыслы со скрытыми переменными — стохастический вариационный вывод (Bartunov et al., 2015).

# ОБЩИЕ ПОДХОДЫ

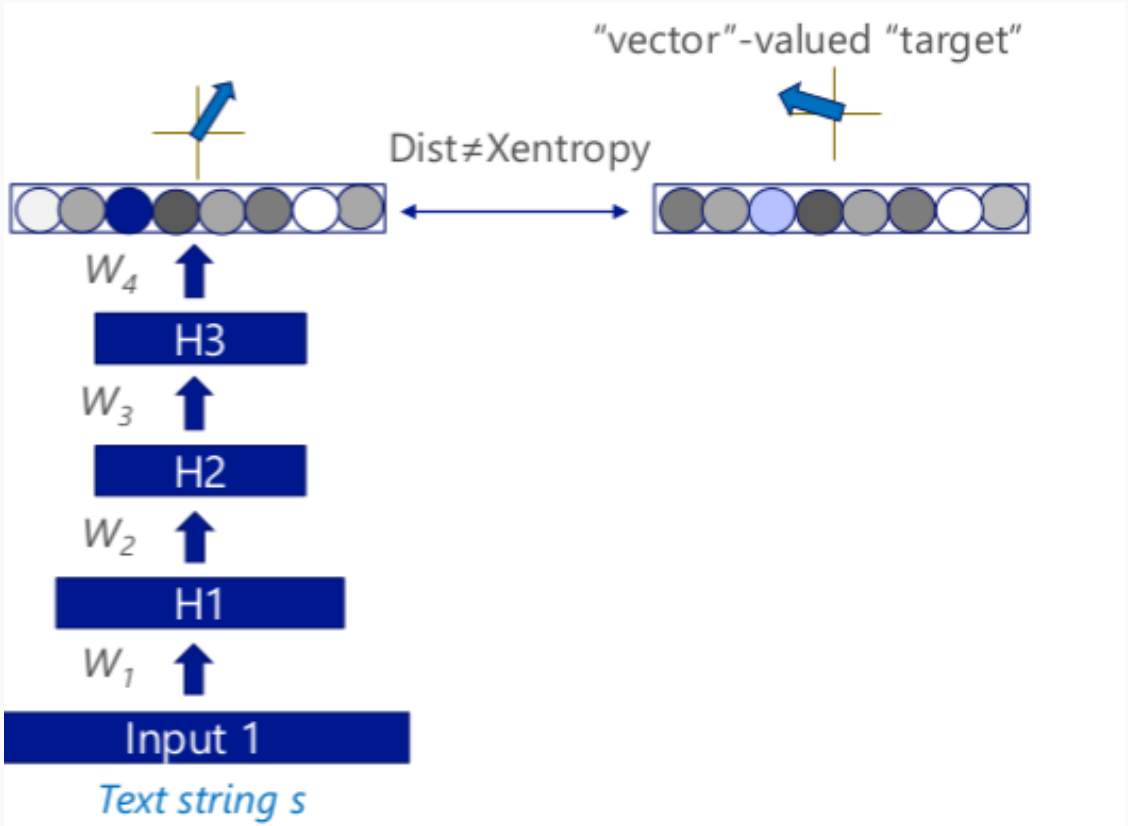
# DSSM

- Ещё один общий подход к NLP на основе CNN – *Deep Structured Semantic Model* (DSSM) (Huang et al., 2013; Gao et al., 2014a; 2014b):
  - классификация (распознавание речи, языковые модели, картинки).



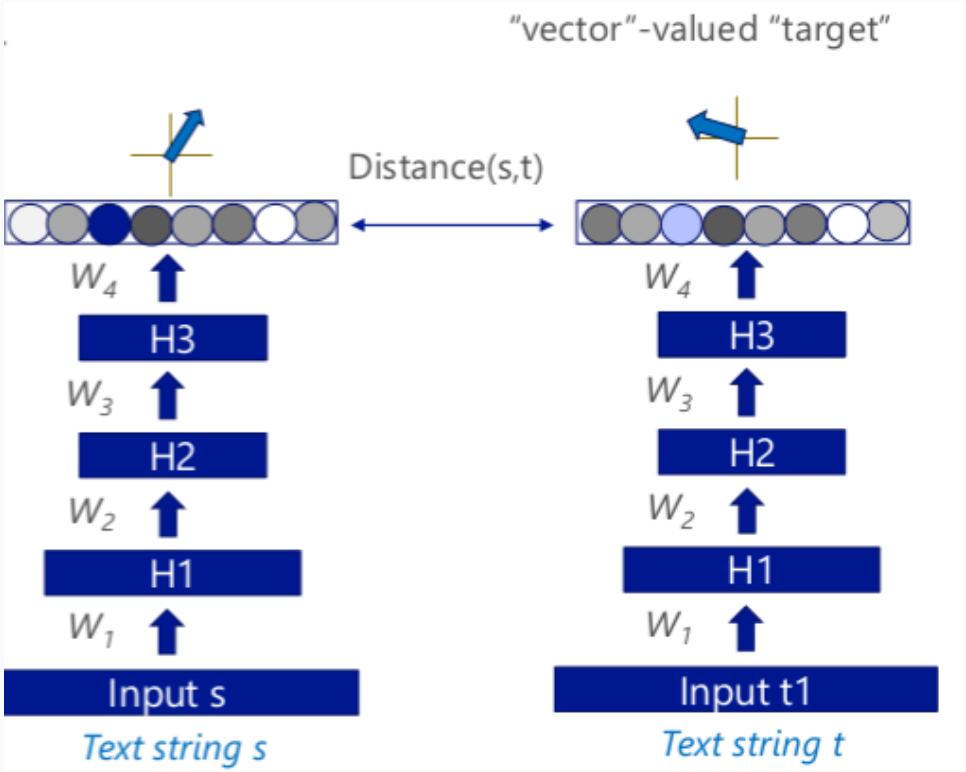
# DSSM

- Ещё один общий подход к NLP на основе CNN – *Deep Structured Semantic Model* (DSSM) (Huang et al., 2013; Gao et al., 2014a; 2014b):
  - векторные правильные ответы для семантической похожести.



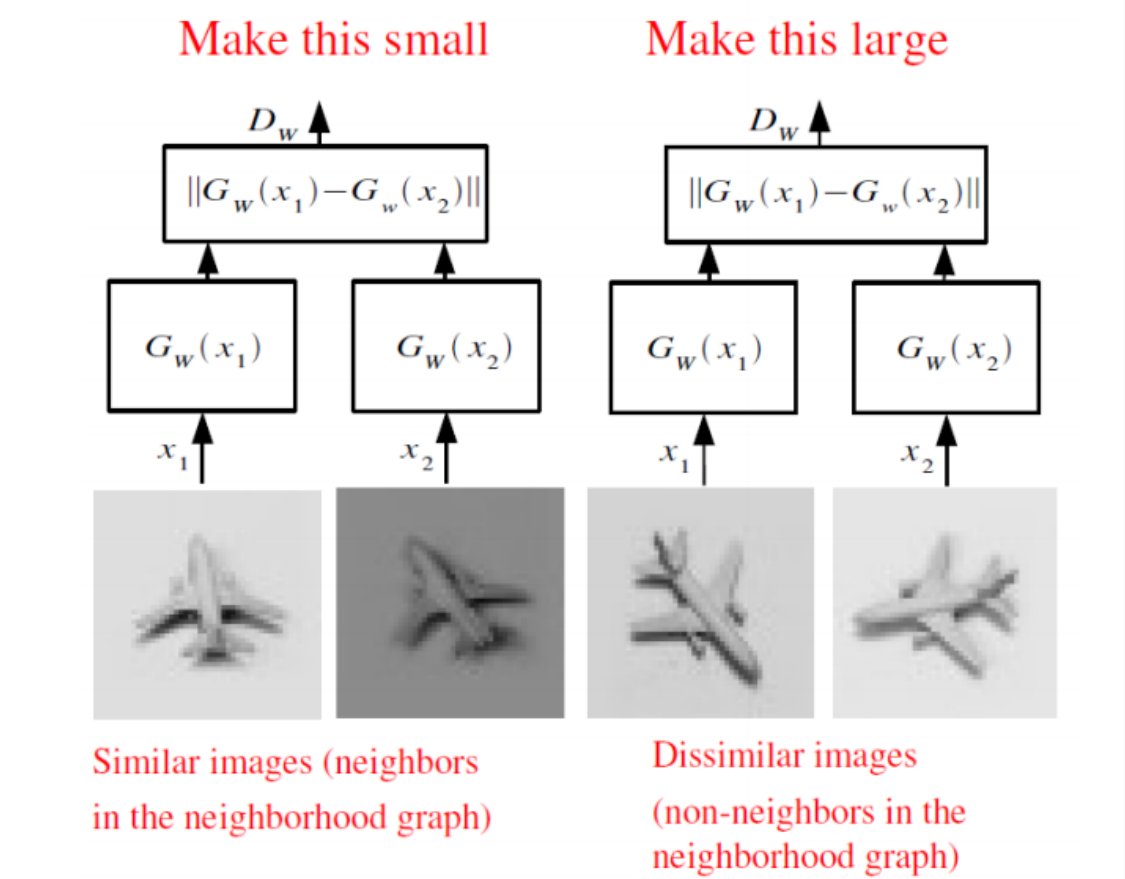
# DSSM

- Ещё один общий подход к NLP на основе CNN – *Deep Structured Semantic Model* (DSSM) (Huang et al., 2013; Gao et al., 2014a; 2014b):
  - чтобы обучить с векторными ответами – принцип отражения (reflection): притягиваем похожие, отталкиваем непохожие.



# DSSM

- Это давняя идея *сиамских сетей* (Bromley et al., 1994)



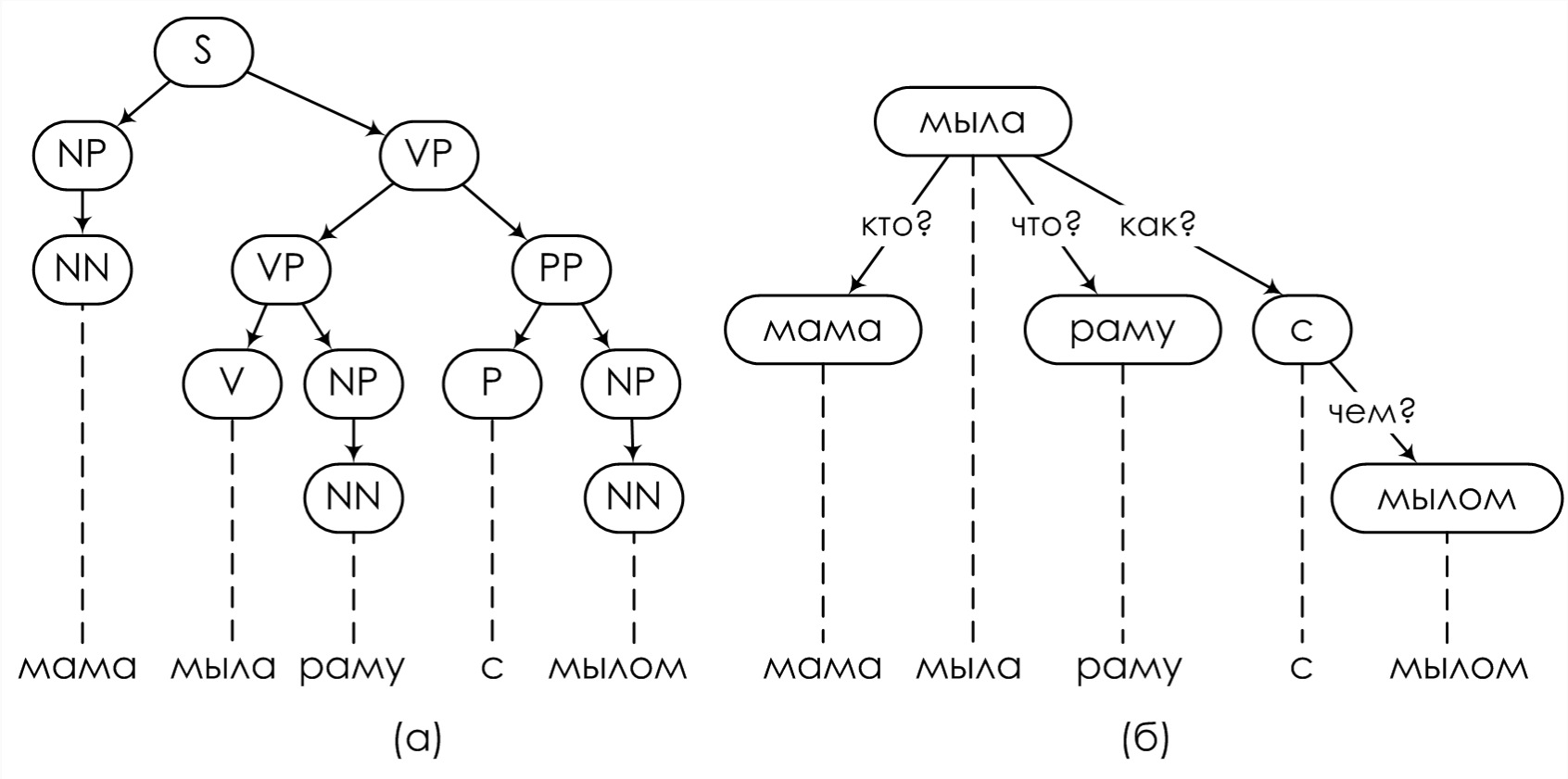
# DSSM

- DSSM можно применять в самых разных контекстах, если есть датасет: семантические embeddings, ответ на вопросы (фактологические), рекомендации и т.д.
- Например, для information retrieval: притягиваем вместе документы и запросы, которым они релевантны, отталкиваем нерелевантные.
- В ноябре 2016 Яндекс выпустил пост о том, что они используют (модифицированный) DSSM в их новом поисковом алгоритме *Палех*.

# Синтаксический разбор

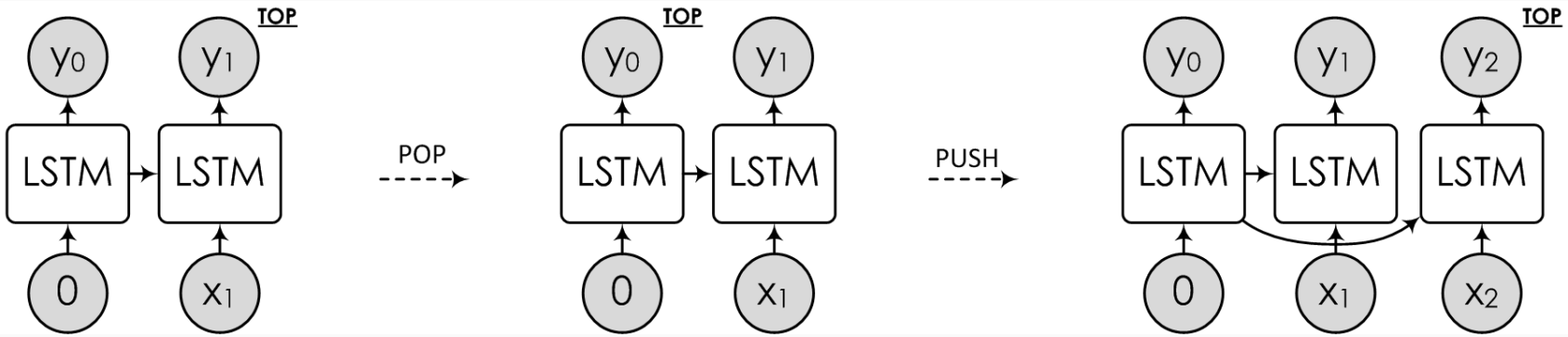
# Синтаксический разбор

- Как же всё-таки построить деревья разбора?



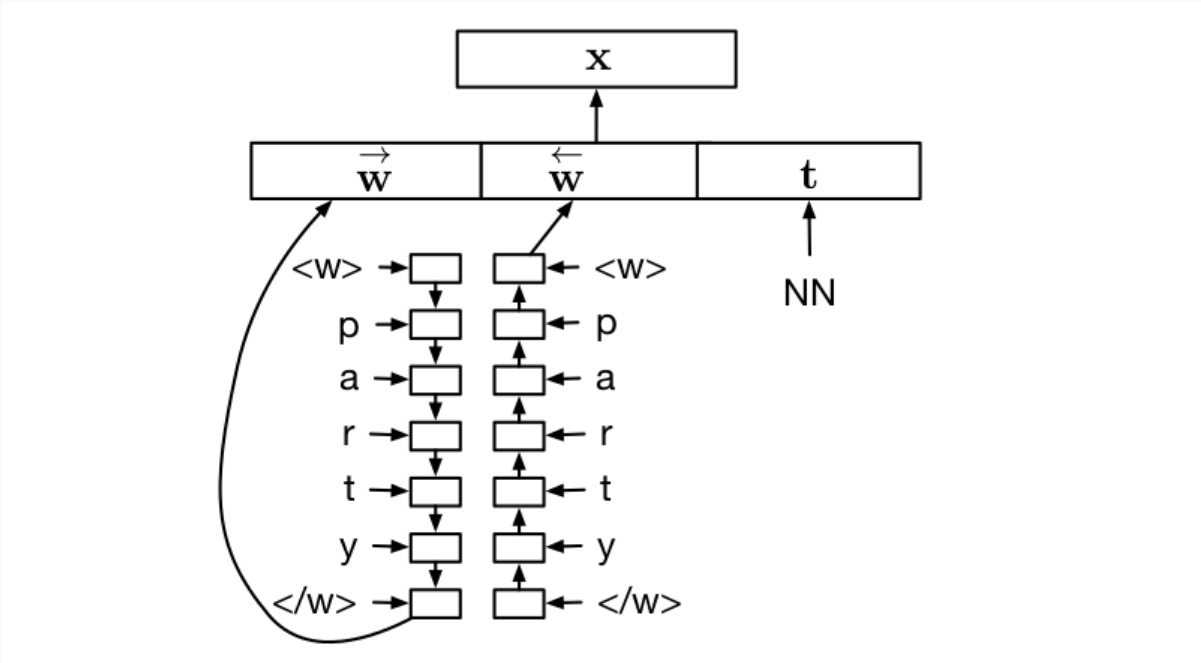
# Синтаксический разбор

- Сейчас обычно *continuous-state parsing*: текущее состояние парсера – вектор из $\mathbb{R}^d$.
- *Stack LSTMs* (Dyer et al., 2015) – парсер управляет тремя структурами данных:
  (1) буфер $B$ содержит последовательность слов;
  (2) стек $S$ хранит частично построенные деревья;
  (3) список $A$ действий, которые уже были предприняты парсером.



# Синтаксический разбор с морфологией

- Важное расширение (Ballesteros et al., 2015):
  - надо учесть морфологию, так что представим слово двунаправленным character-level LSTM;
  - результаты становятся лучше на ряде языков (а русского почему-то нет – дерзайте).



# Метрики качества для sequence-to-sequence моделей

- Дальше будет самое интересное: машинный перевод, диалоговые модели, ответ на вопросы.
- Но как мы будем оценивать NLP-модели, которые генерируют текст?
- Есть метрики качества, которые сравнивают результат с правильными ответами:
  - BLEU (Bilingual Evaluation Understudy): перевзвешенная precision (в т.ч. для нескольких правильных ответов);
  - METEOR: гармоническое среднее precision и recall по униграммам;
  - TER (Translation Edit Rate): число исправлений между выходом и правильным ответом, делённое на среднее число слов;
  - LEPOR: комбинируем базовые факторы и метрики с настраиваемыми параметрами.
- Есть ещё куча метрик, связанных с представлениями слов и предложений (хотим поближе к правильному ответу).
- Одна только проблема...

# Метрики качества для sequence-to-sequence моделей

- ...всё это вообще не работает.

| | Twitter | | | | Ubuntu | | | |
|---|---|---|---|---|---|---|---|---|
| Metric | **Spearman** | p-value | **Pearson** | p-value | **Spearman** | p-value | **Pearson** | p-value |
| Greedy | 0.2119 | 0.034 | 0.1994 | 0.047 | 0.05276 | 0.6 | 0.02049 | 0.84 |
| Average | 0.2259 | 0.024 | 0.1971 | 0.049 | -0.1387 | 0.17 | -0.1631 | 0.10 |
| Extrema | 0.2103 | 0.036 | 0.1842 | 0.067 | 0.09243 | 0.36 | -0.002903 | 0.98 |
| METEOR | 0.1887 | 0.06 | 0.1927 | 0.055 | 0.06314 | 0.53 | 0.1419 | 0.16 |
| BLEU-1 | 0.1665 | 0.098 | 0.1288 | 0.2 | -0.02552 | 0.8 | 0.01929 | 0.85 |
| BLEU-2 | 0.3576 | $< 0.01$ | 0.3874 | $< 0.01$ | 0.03819 | 0.71 | 0.0586 | 0.56 |
| BLEU-3 | 0.3423 | $< 0.01$ | 0.1443 | 0.15 | 0.0878 | 0.38 | 0.1116 | 0.27 |
| BLEU-4 | 0.3417 | $< 0.01$ | 0.1392 | 0.17 | 0.1218 | 0.23 | 0.1132 | 0.26 |
| ROUGE | 0.1235 | 0.22 | 0.09714 | 0.34 | 0.05405 | 0.5933 | 0.06401 | 0.53 |
| Human | 0.9476 | $< 0.01$ | 1.0 | 0.0 | 0.9550 | $< 0.01$ | 1.0 | 0.0 |

Table 3: Correlation between each metric and human judgements for each response. Correlations shown in the human row result from randomly dividing human judges into two groups.

- Тут нужно что-то новое.

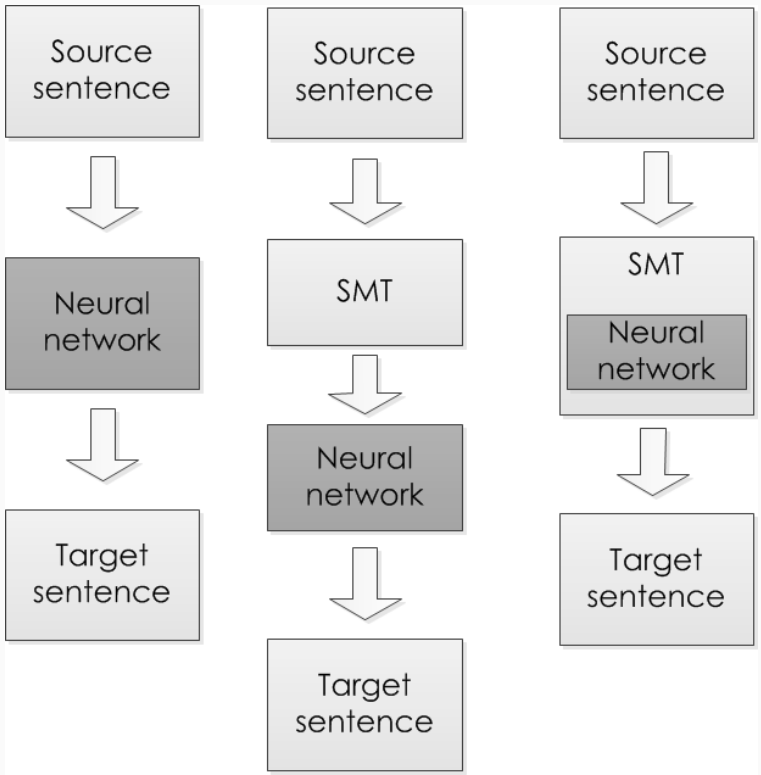
# Машинный перевод: encoder-decoder и внимание

# Машинный перевод

- Перевод – очень хорошая задача:
  - очевидно очень практическая;
  - очевидно очень высокоуровневая, требует понимания;
  - считается довольно неплохо квантифицируемой (BLEU, TER – хотя см. выше);
  - имеет большие доступные датасеты параллельных переводов.

# МАШИННЫЙ ПЕРЕВОД

- Статистический машинный перевод (statistical machine translation, SMT): моделируем условную вероятность $p(y \mid x)$ перевода $y$ при условии исходного текста $x$.
- Классический SMT: моделируем $\log p(y \mid x)$ линейной комбинацией признаков, строим признаки.



# МАШИННЫЙ ПЕРЕВОД

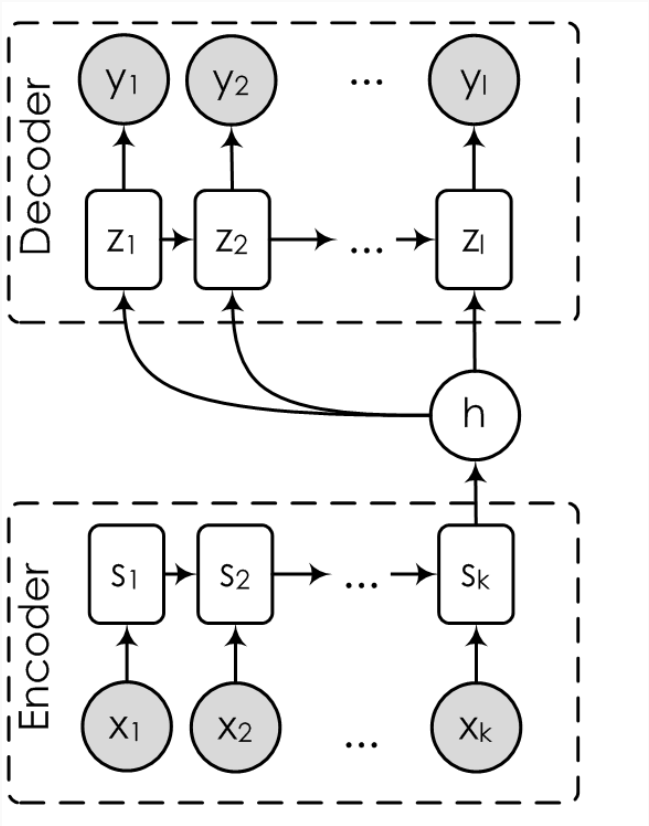
- Нам больше интересно моделирование sequence-to-sequence:
  - RNN естественным образом моделирует последовательность $X = (x_1, x_2, \dots, x_T)$ как $p(x_1)$, $p(x_2 \mid x_1)$, ..., $p(x_T \mid x_{<T}) = p(x_T \mid x_{T-1}, \dots, x_1)$, и теперь $p(X)$ – это просто

$$p(X) = p(x_1)p(x_2 \mid x_1) \dots p(x_k \mid x_{<k}) \dots p(x_T \mid x_{<T});$$

  - так RNN и в языковых моделях используются;
  - предсказываем следующее слово на основе скрытого состояния и предыдущего слова;
- Как применить эту идею к переводу?

# Encoder-decoder архитектуры

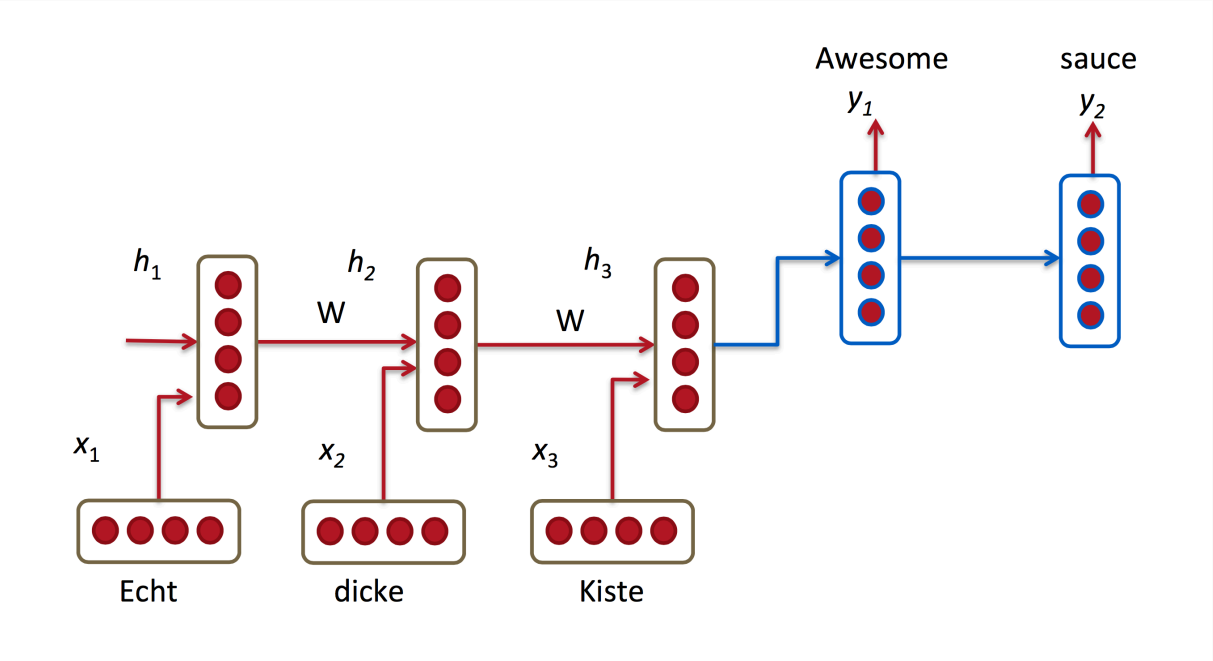
- Encoder-decoder архитектуры (Sutskever et al., 2014; Cho et al., 2014):




- Сначала кодируем, потом декодируем обратно.

# ENCODER-DECODER АРХИТЕКТУРЫ

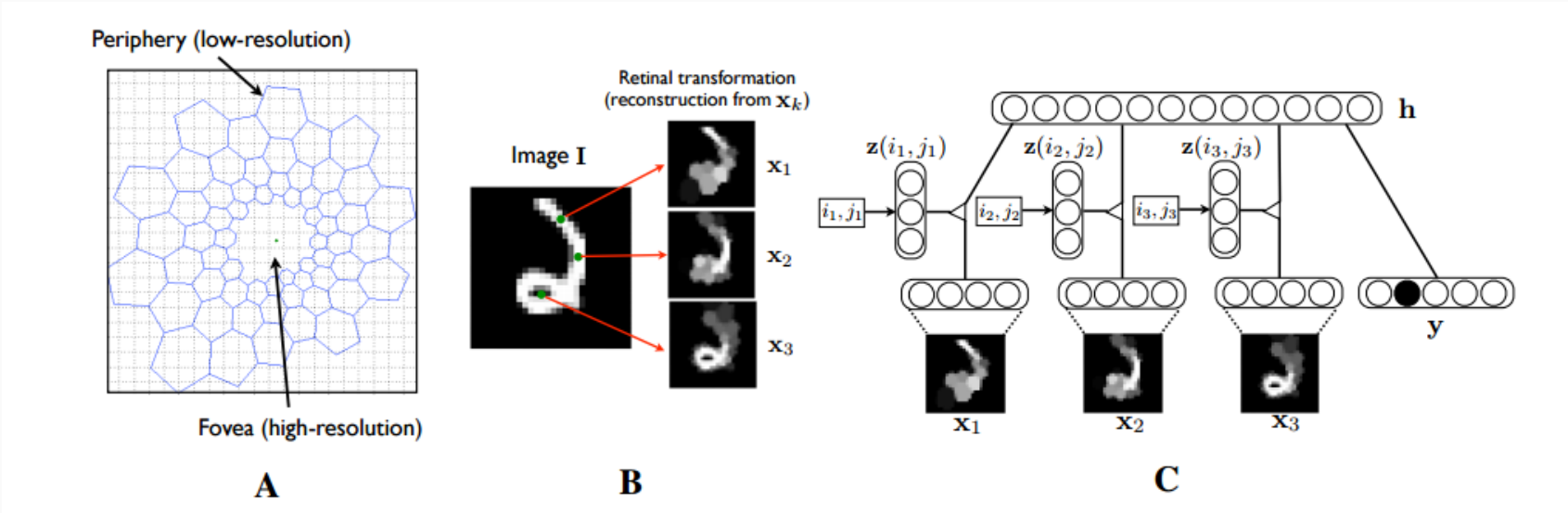
- Так же может работать и с переводом.




- Проблема: надо сжимать всё предложение в один вектор.
- С длинными участками текста это вообще перестаёт работать.

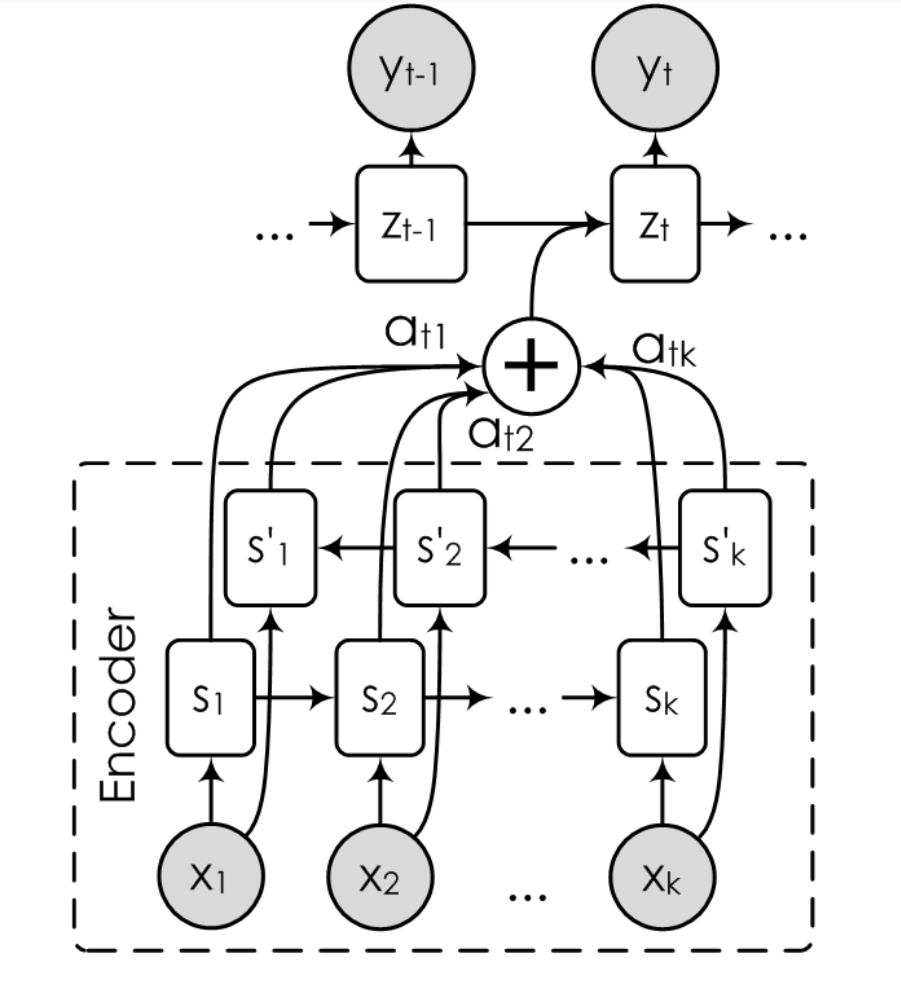
# ВНИМАНИЕ В НЕЙРОННЫХ СЕТЯХ

- Решение: давайте обучим специальные веса, показывающие, насколько та или иная часть входа важна для текущего выхода.
- Это чем-то напоминает механизм *внимания* у людей: что помещать в рабочую память, а что не надо.
- Первые применения в нейросетях – foveal glimpses на RBM (Larochelle, Hinton, 2010)



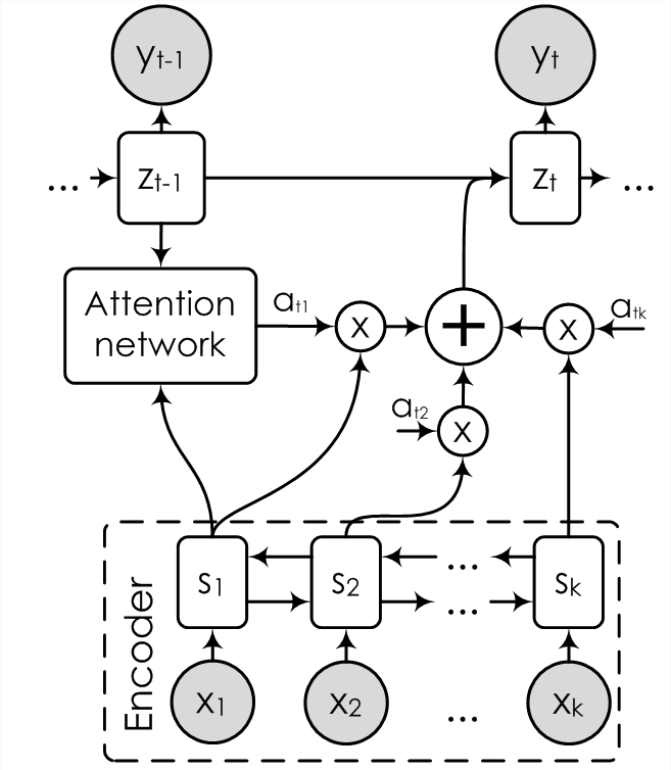
# ВНИМАНИЕ В НЕЙРОННЫХ СЕТЯХ

- Прямое применение этой идеи – двунаправленный LSTM плюс внимание (Bahdanau et al. 2014):



# ВНИМАНИЕ В НЕЙРОННЫХ СЕТЯХ

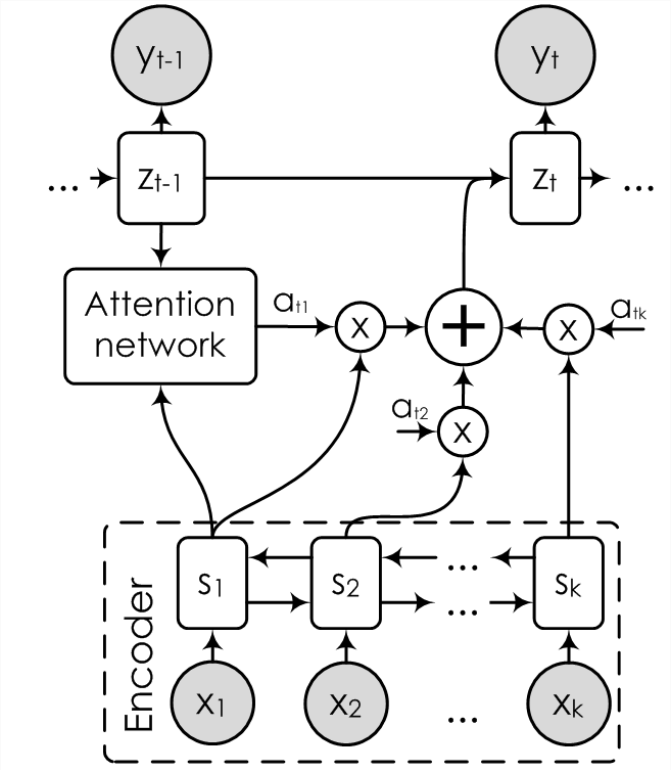
- *Мягкое внимание* (soft attention) (Luong et al. 2015a; 2015b; Jean et al. 2015):
  - encoder – двунаправленная RNN, есть оба контекста;
  - сеть внимания выдаёт оценку релевантности – надо ли переводить это слово прямо сейчас?



# ВНИМАНИЕ В НЕЙРОННЫХ СЕТЯХ

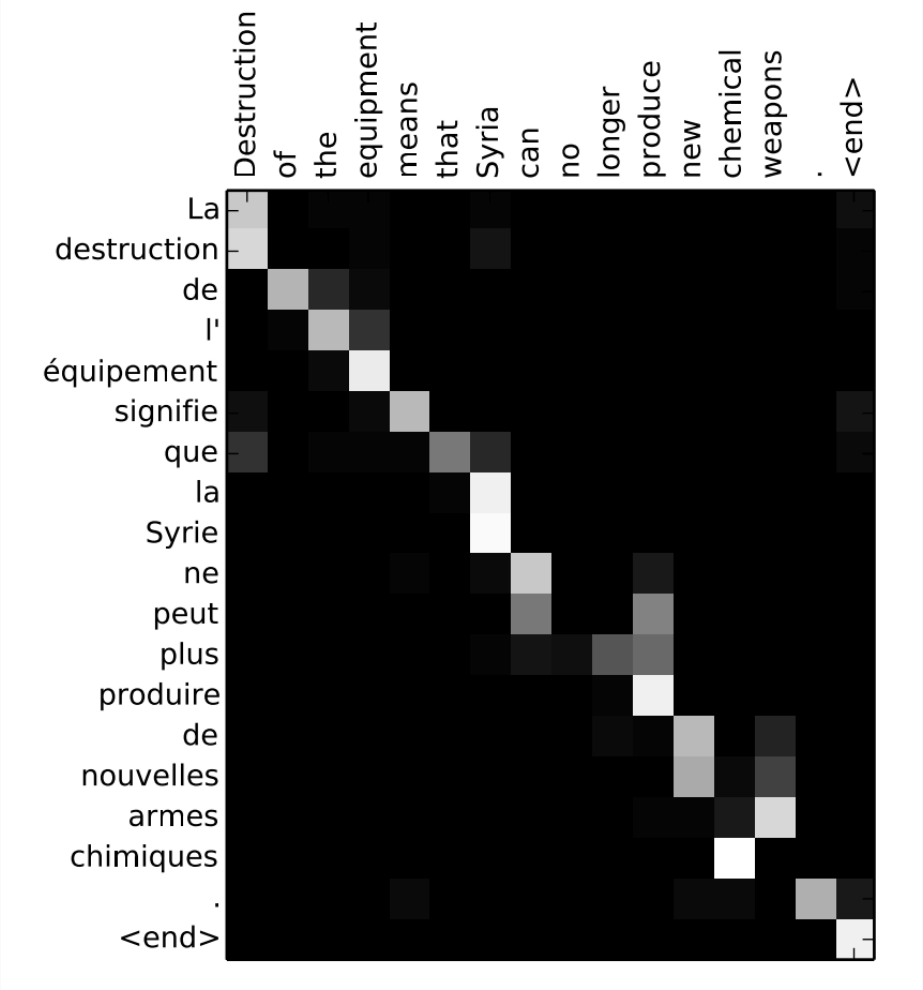
- Формально очень просто: считаем веса внимания $\alpha_{tj}$ и перевзвешиваем векторы контекстов:

$$e_{tj} = a(z_{t-1}, j), \quad \alpha_{tj} = \mathrm{softmax}(e_{tj}; e_{t*}),$$
$$c_t = \sum_j \alpha_{tj} h_j, \text{ и теперь } z_t = f(s_{t-1}, y_{t-1}, c_i).$$



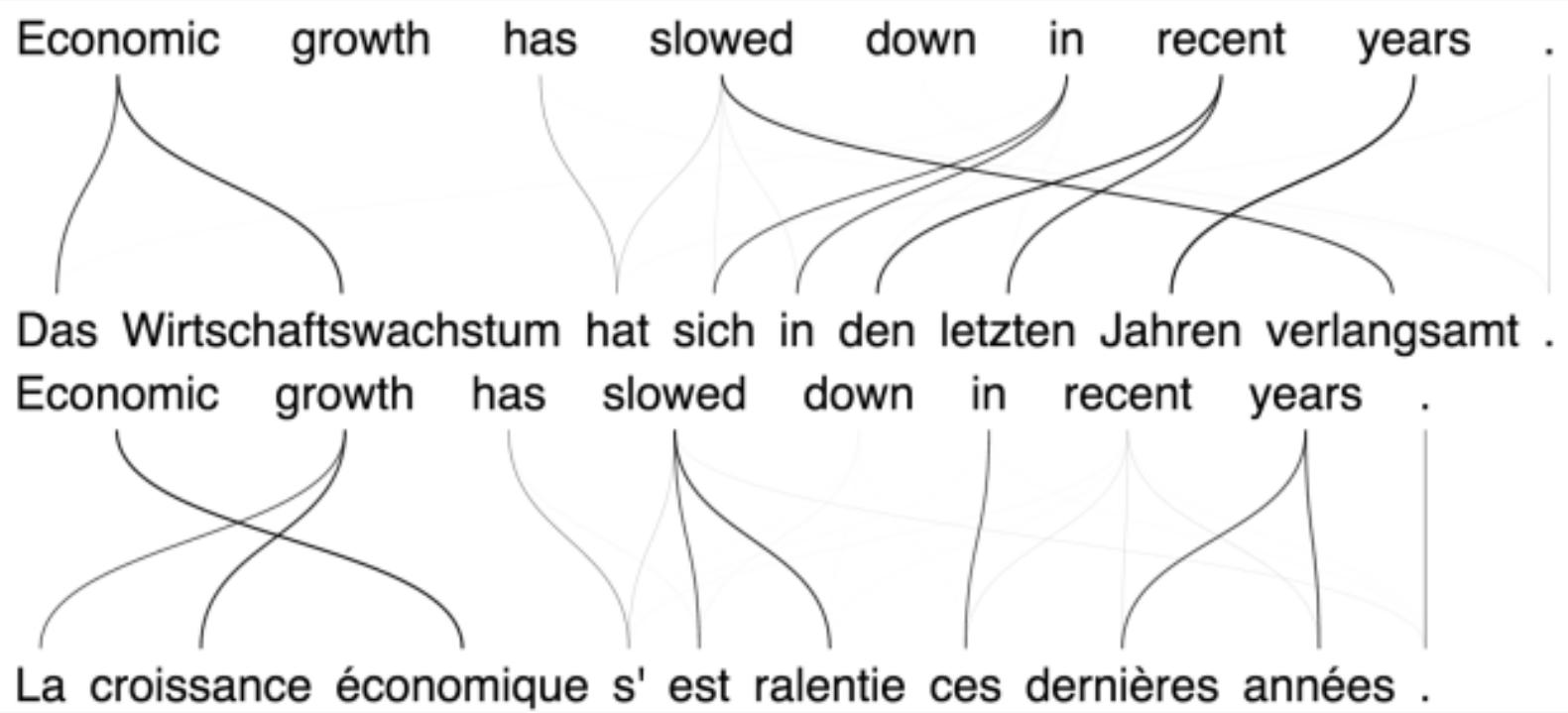
# ВНИМАНИЕ В НЕЙРОННЫХ СЕТЯХ

- В результате можно визуализировать, на что смотрит сеть:



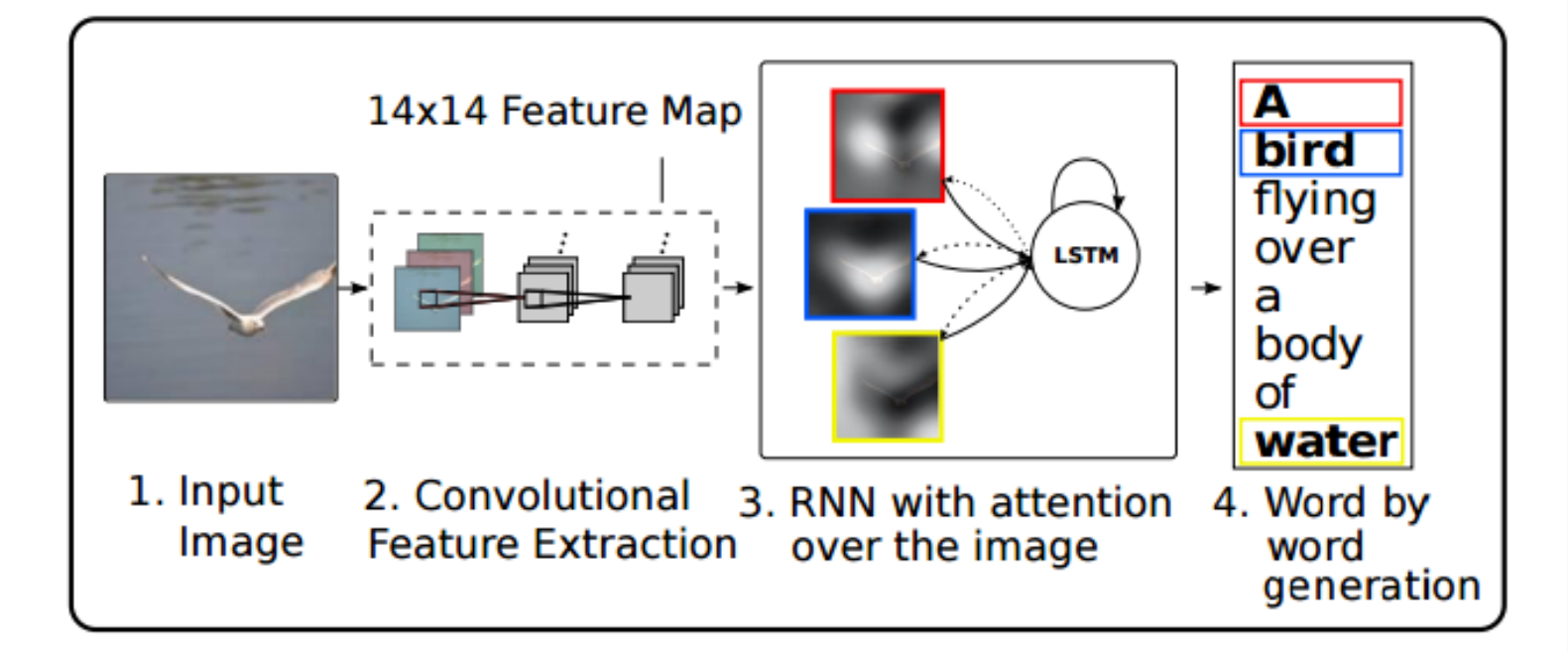
# ВНИМАНИЕ В НЕЙРОННЫХ СЕТЯХ

- Получается гораздо лучше порядок слов:



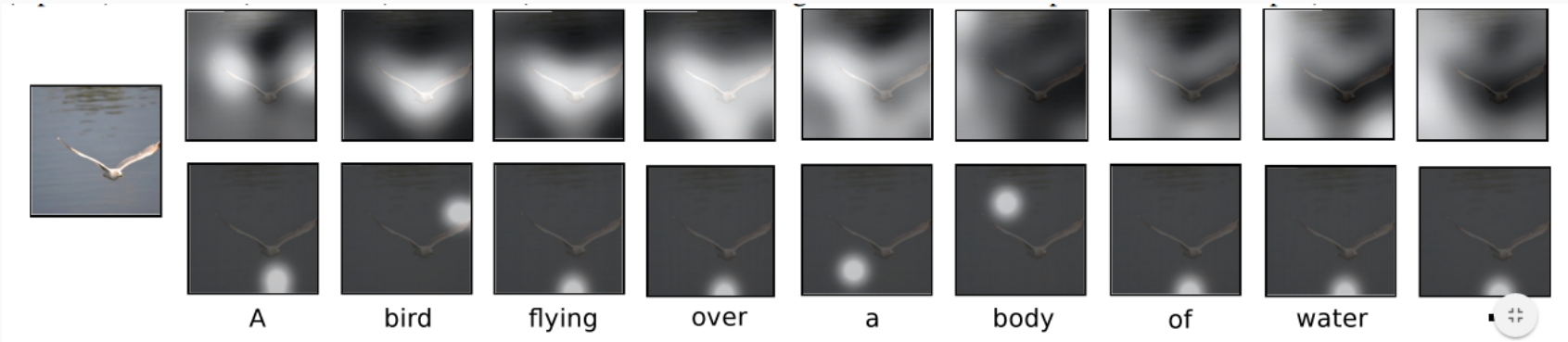
# ВНИМАНИЕ В НЕЙРОННЫХ СЕТЯХ

- Конечно, есть и другие применения механизмов внимания.
- Show, Attend, and Tell (Xu et al., 2015): генерируем подписи к рисункам.



# ВНИМАНИЕ В НЕЙРОННЫХ СЕТЯХ

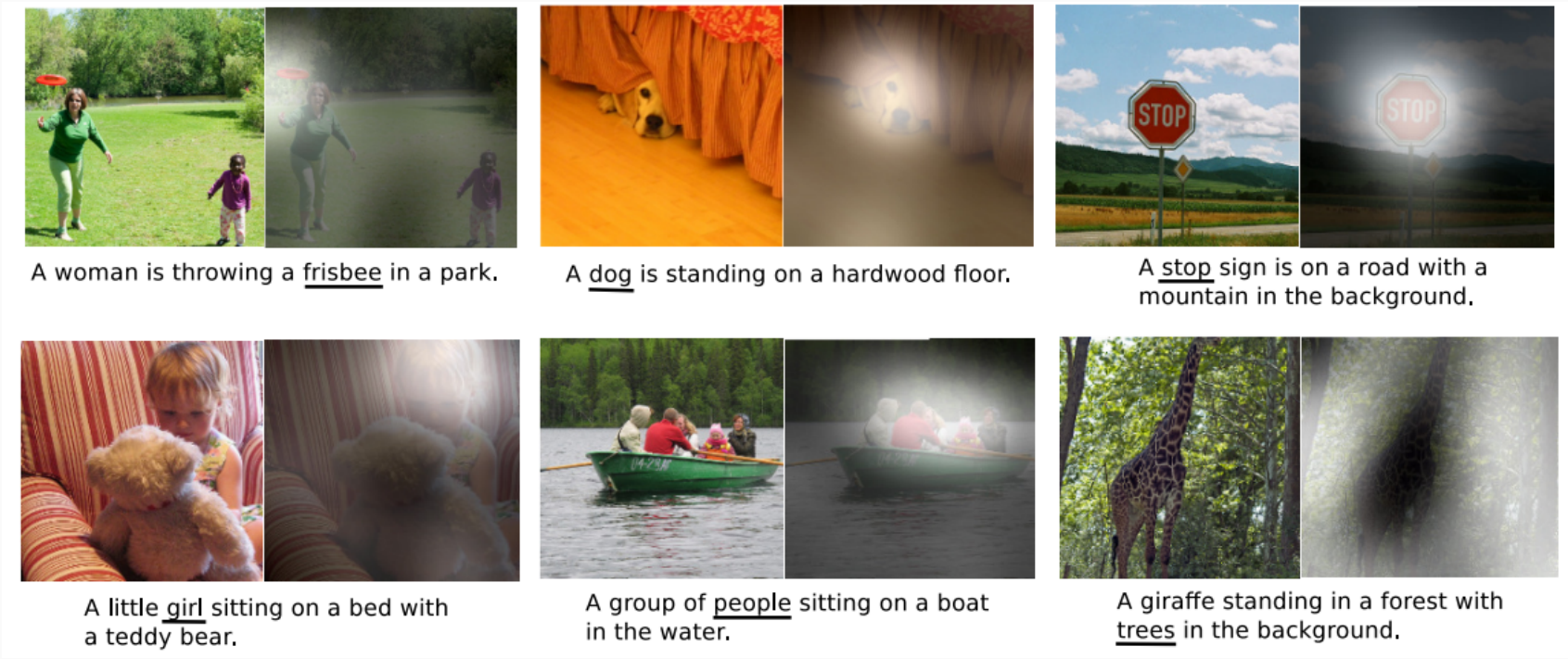
- Soft attention vs. hard attention (стохастически выбираем однозначный кусок картинки).




- Hard attention обучается максимизацией вариационной нижней оценки (об этом позже).

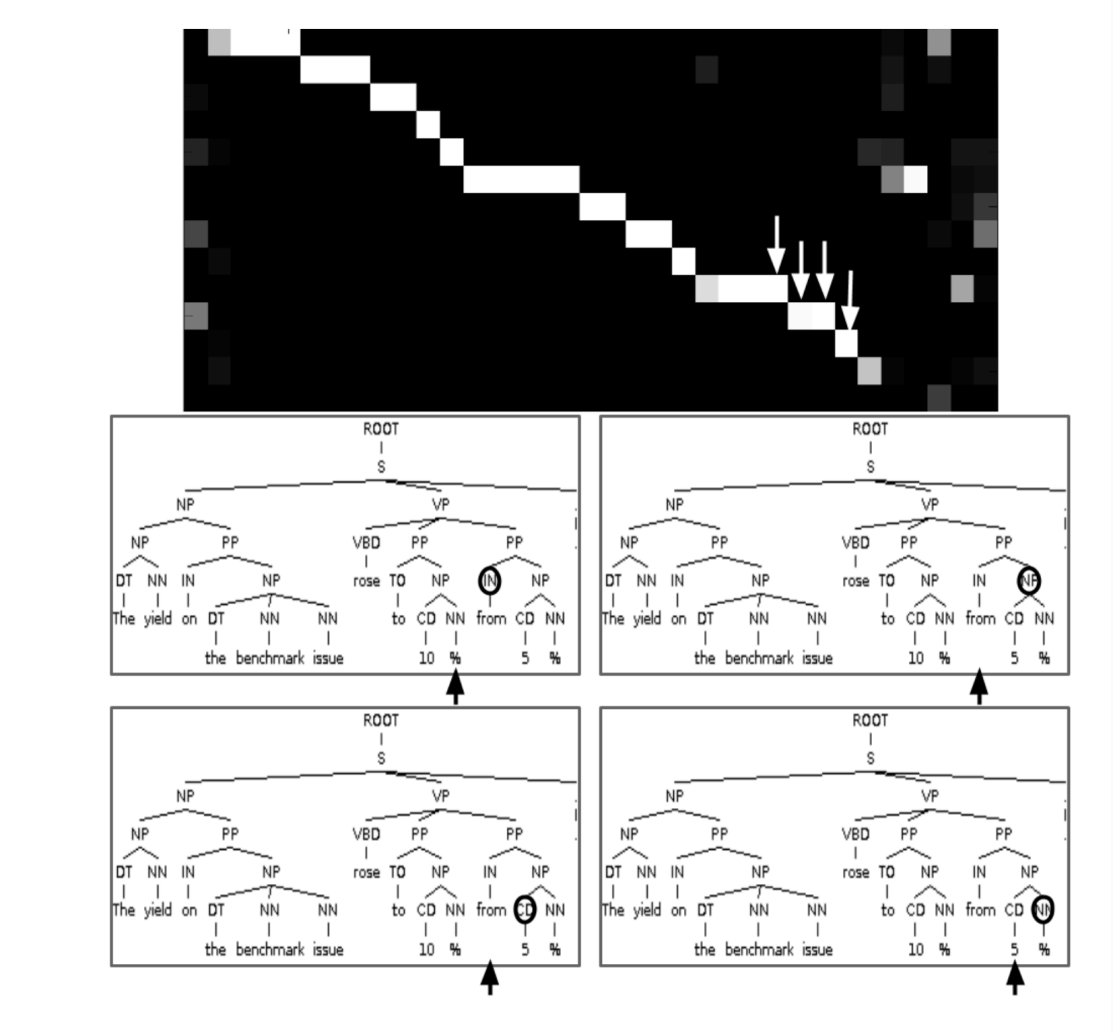
# ВНИМАНИЕ В НЕЙРОННЫХ СЕТЯХ

- Часто получаются очень хорошие результаты:



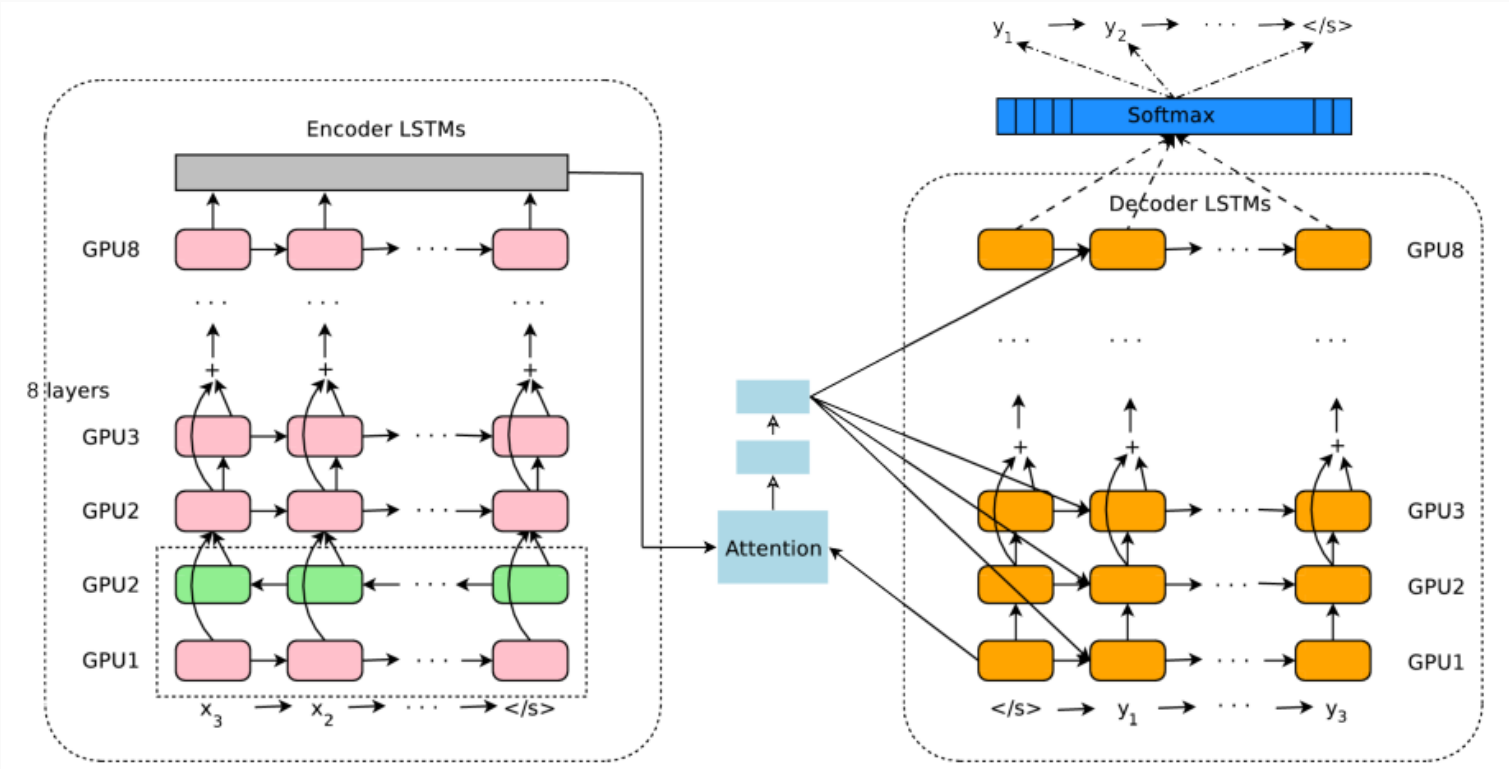
# ВНИМАНИЕ В НЕЙРОННЫХ СЕТЯХ

- Ещё ближе к теме – «Grammar as a Foreign Language» (Vinyals et al., 2015)



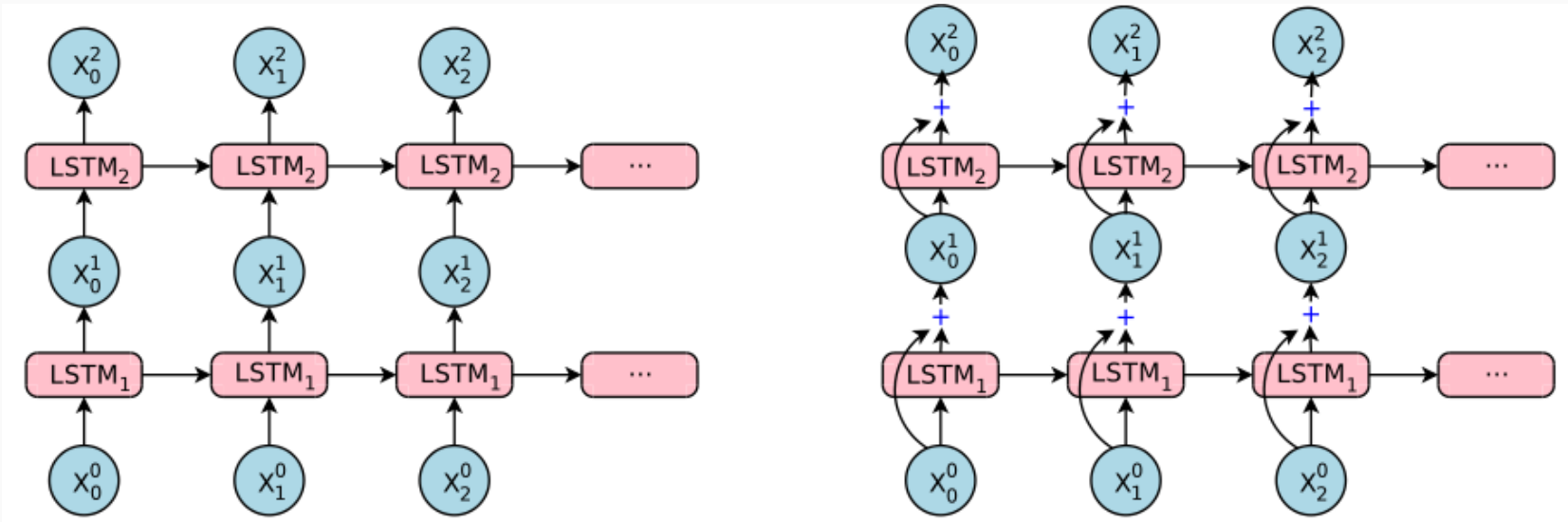
# GOOGLE TRANSLATE

- Сентябрь 2016: Wu et al., *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*:
  - как на самом деле работает Google Translate;
  - базовая архитектура та же самая: encoder, decoder, attention;
  - RNN глубокие, по 8 уровней в encoder и decoder:

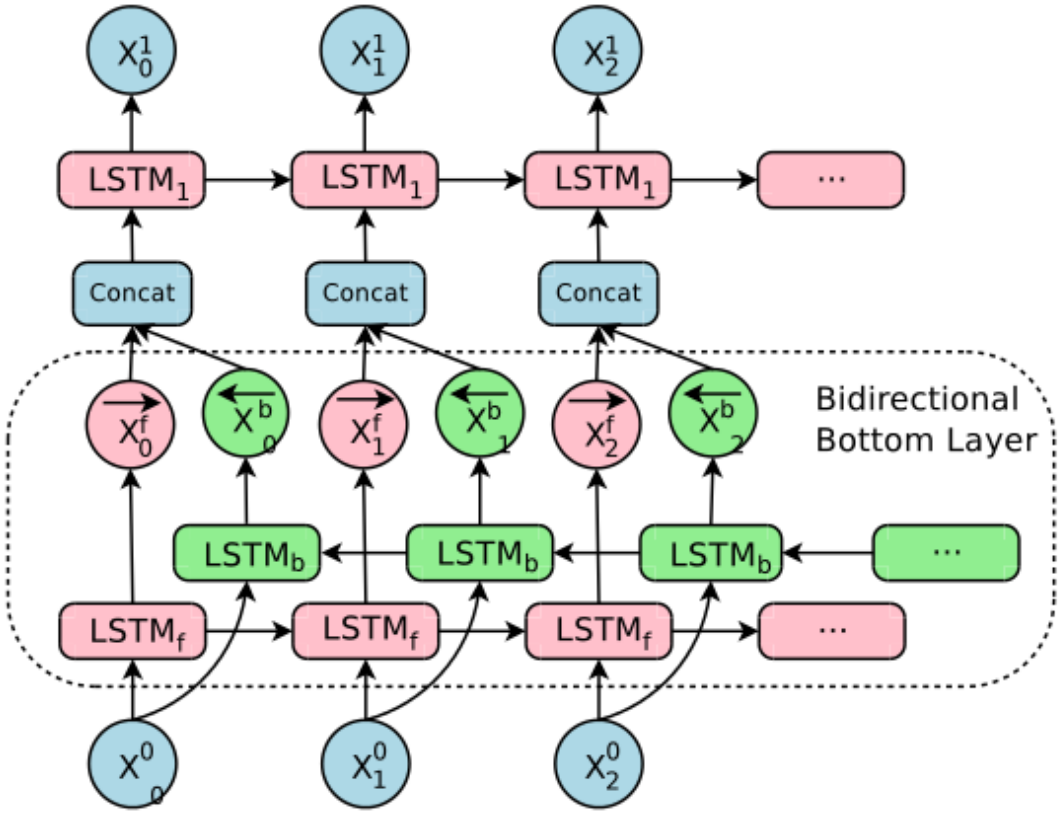


# GOOGLE TRANSLATE

- Сентябрь 2016: Wu et al., *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*:
  - просто stacked LSTM перестают работать далее 4-5 уровней;
  - поэтому добавляют остаточные связи, как в ResNet:



- Сентябрь 2016: Wu et al., *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*:
  - нижний уровень, естественно, двунаправленный:



# GOOGLE TRANSLATE

- Сентябрь 2016: Wu et al., *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*:
- в GNMT ещё две идеи о сегментации слов:
  - *wordpiece model*: разбить слова на кусочки (отдельной моделью); пример из статьи:

    `Jet makers feud over seat width with big orders at stake`

    превращается в

    `_J et _makers _fe ud _over _seat _width _with _big _orders _at _stake`

  - *mixed word/character model*: конвертировать слова, не попадающие в словарь, в последовательность букв-токенов; пример из статьи:

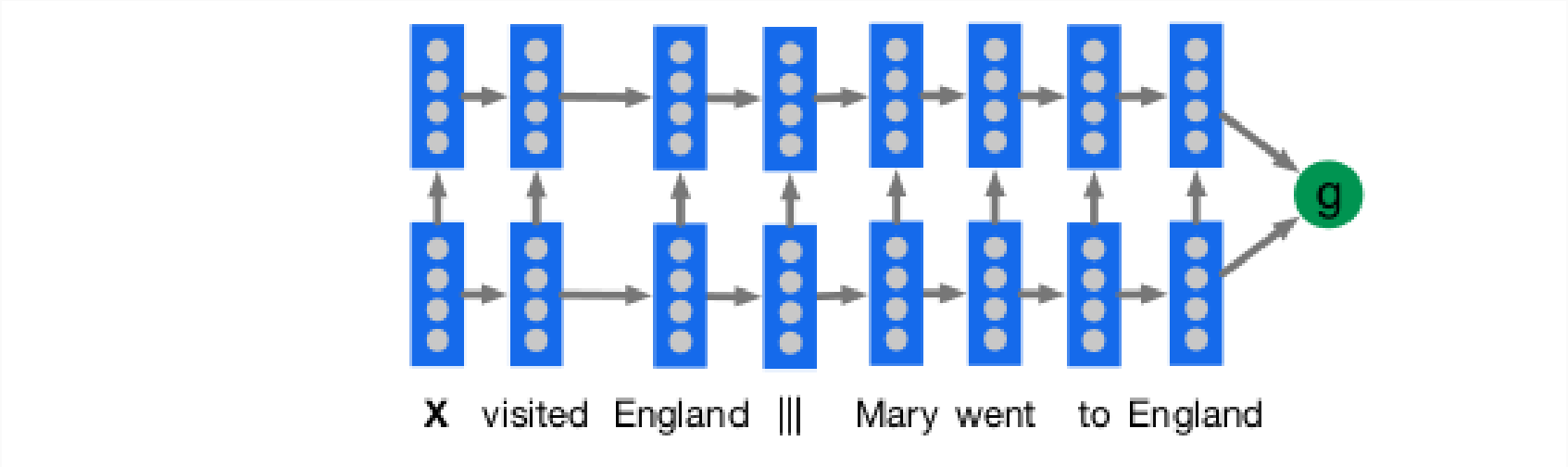
    `Miki` превращается в `<B>M <M>i <M>k <E>i`

# TEACHING MACHINES TO READ

- (Hermann et al., 2015): «Teaching machines to read and comprehend» (Google DeepMind)
- Предлагают новый способ построить датасет для понимания, автоматически создавая тройки (context, query, answer) из текстов новостей и т.п.

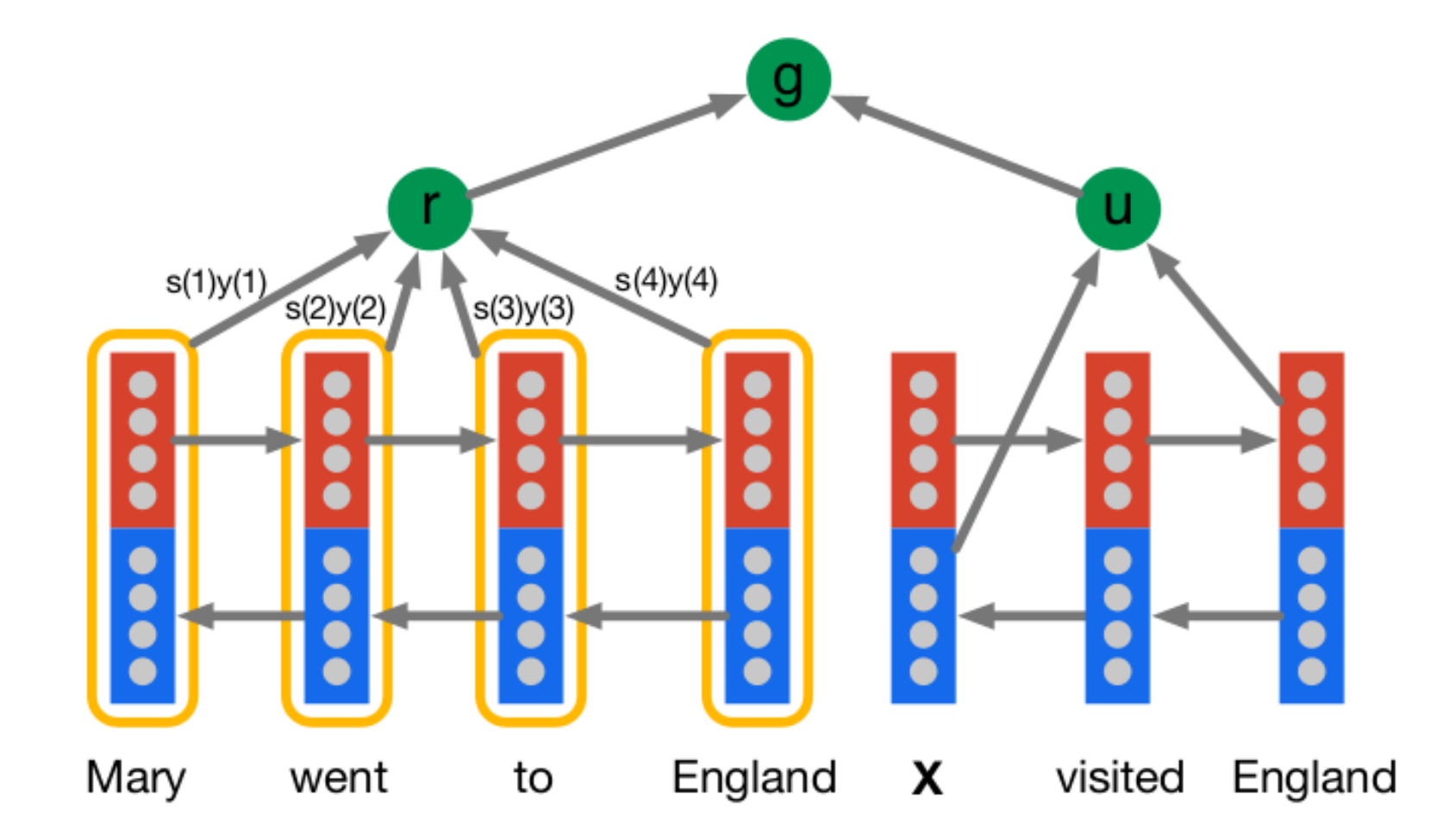
| Original Version | Anonymised Version |
|---|---|
| **Context** | |
| The BBC producer allegedly struck by Jeremy Clarkson will not press charges against the "Top Gear" host, his lawyer said Friday. Clarkson, who hosted one of the most-watched television shows in the world, was dropped by the BBC Wednesday after an internal investigation by the British broadcaster found he had subjected producer Oisin Tymon "to an unprovoked physical and verbal attack." ... | the *ent381* producer allegedly struck by *ent212* will not press charges against the " *ent153* " host , his lawyer said friday . *ent212* , who hosted one of the most - watched television shows in the world , was dropped by the *ent381* wednesday after an internal investigation by the *ent180* broadcaster found he had subjected producer *ent193* " to an unprovoked physical and verbal attack . " ... |
| **Query** | |
| Producer **X** will not press charges against Jeremy Clarkson, his lawyer says. | producer **X** will not press charges against *ent212* , his lawyer says . |
| **Answer** | |
| Oisin Tymon | *ent193* |

- Базовая модель – глубокий LSTM:



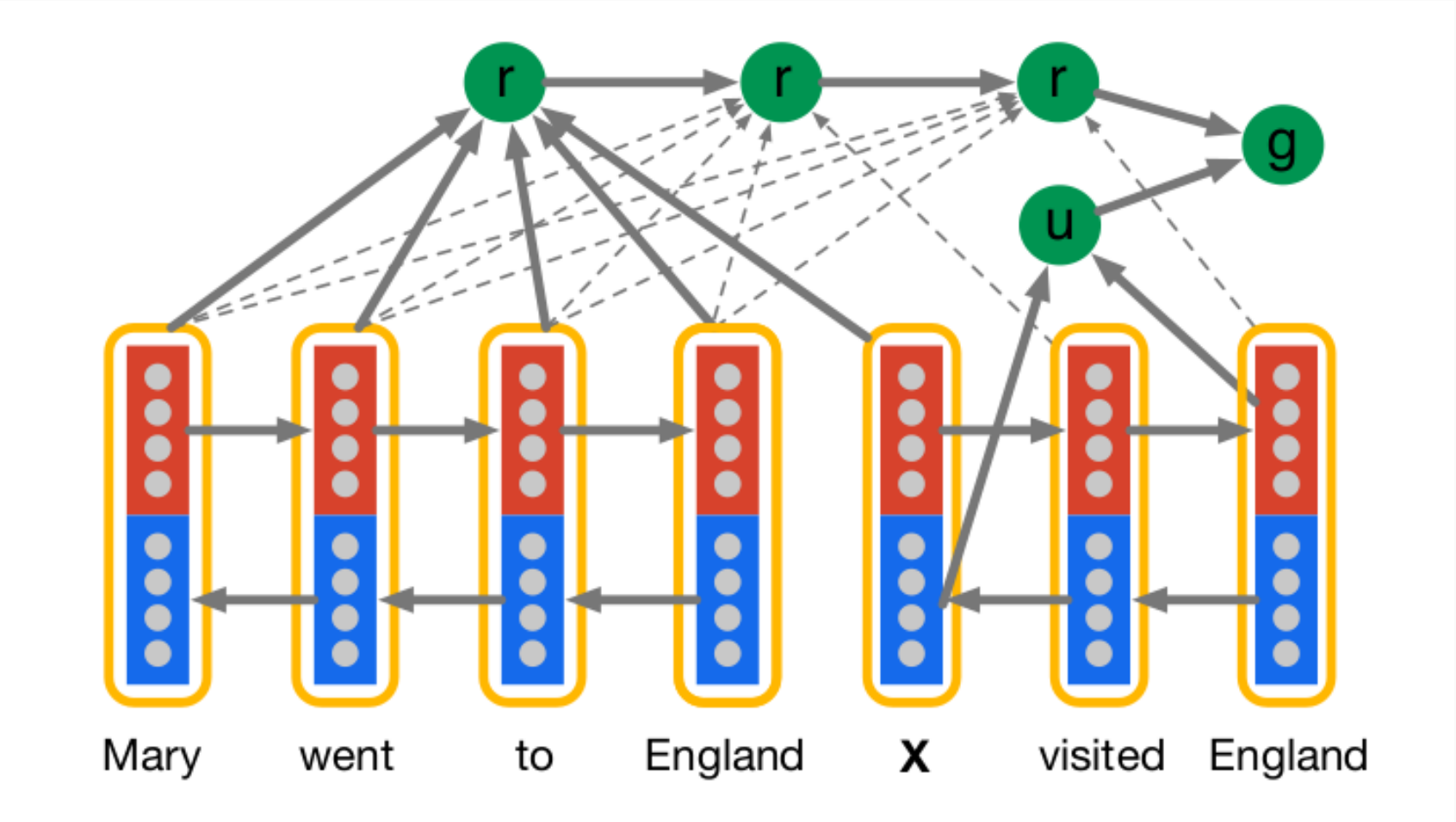

- Но так, конечно, совсем плохо работает.

- Attentive Reader: обучаемся, на какую часть документа смотреть

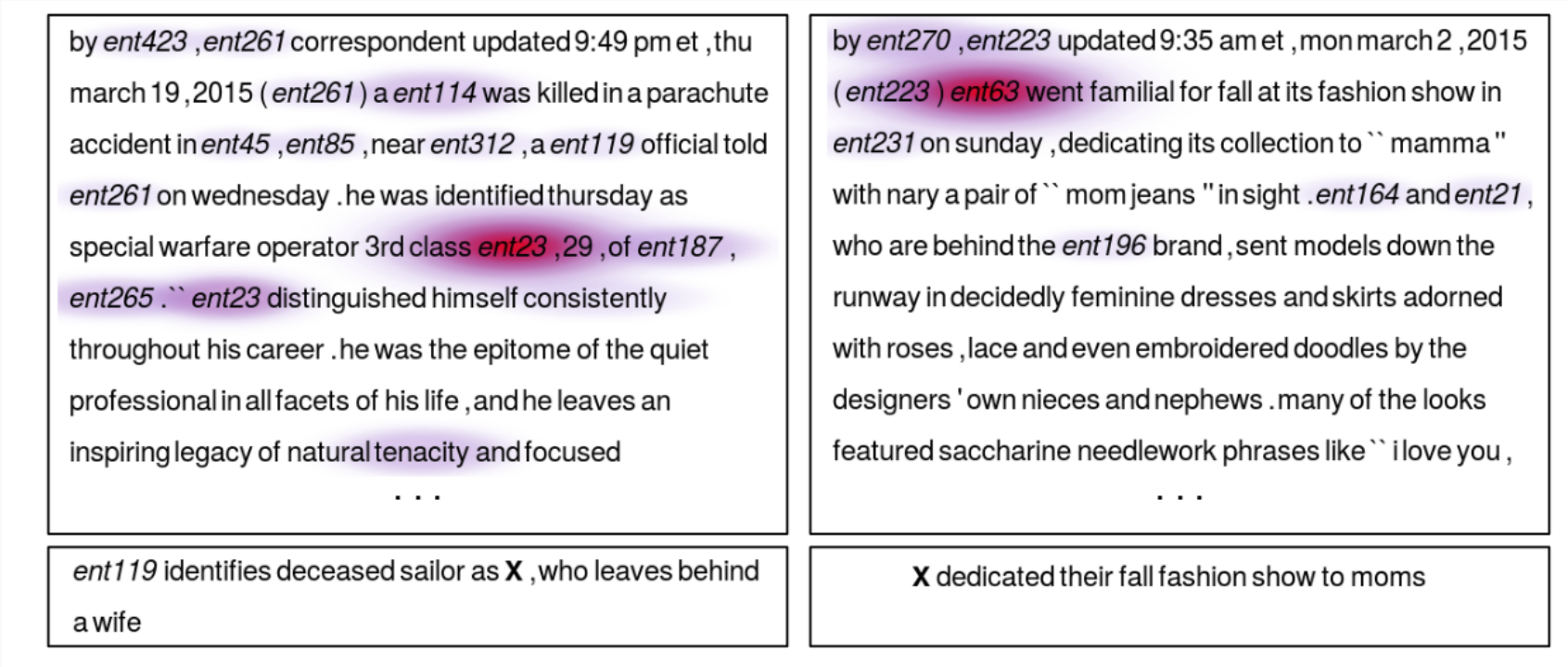


# TEACHING MACHINES TO READ

- Impatient Reader: можем перечитывать нужные части документа по мере прочтения запроса

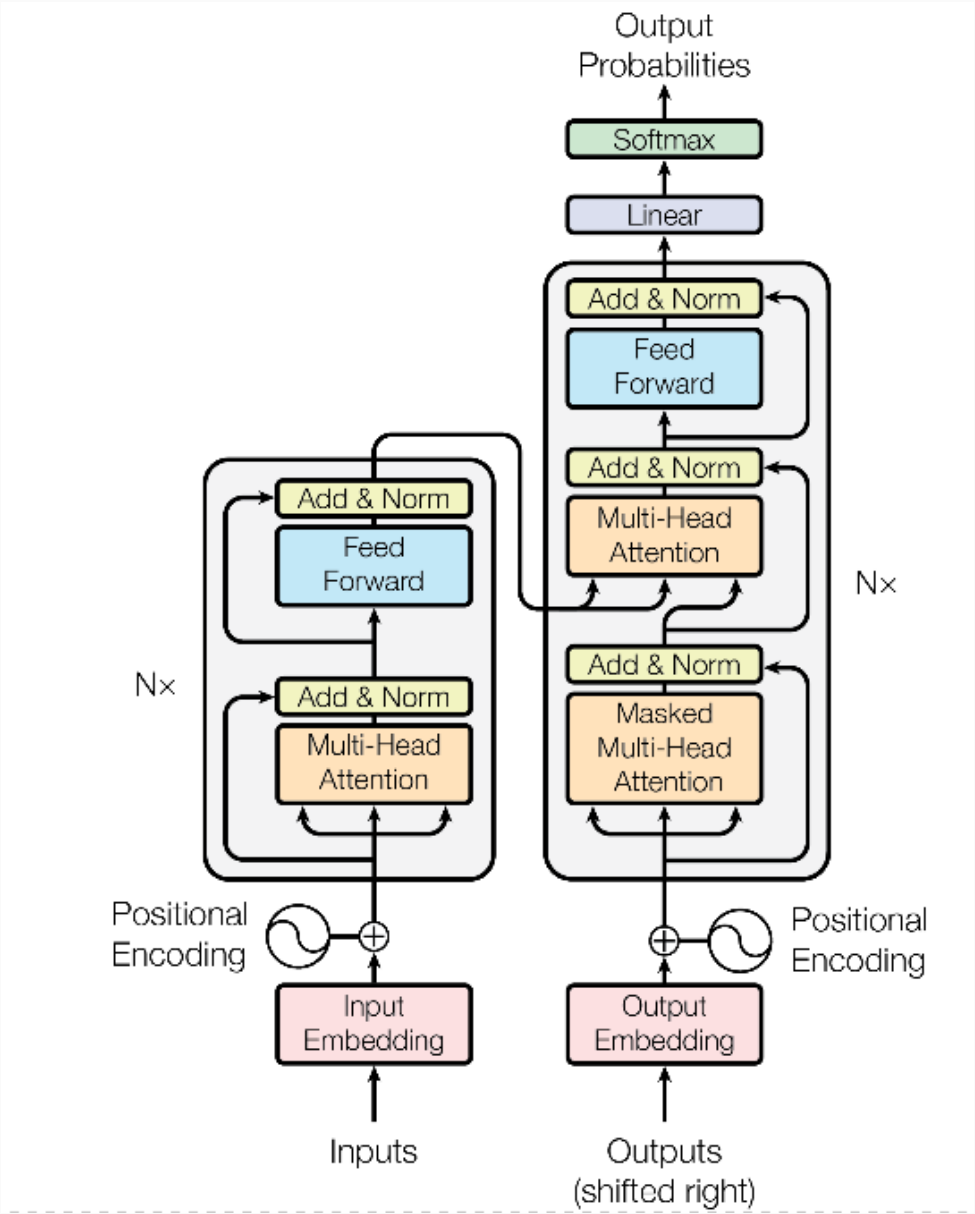


- Получаются разумные карты внимания:



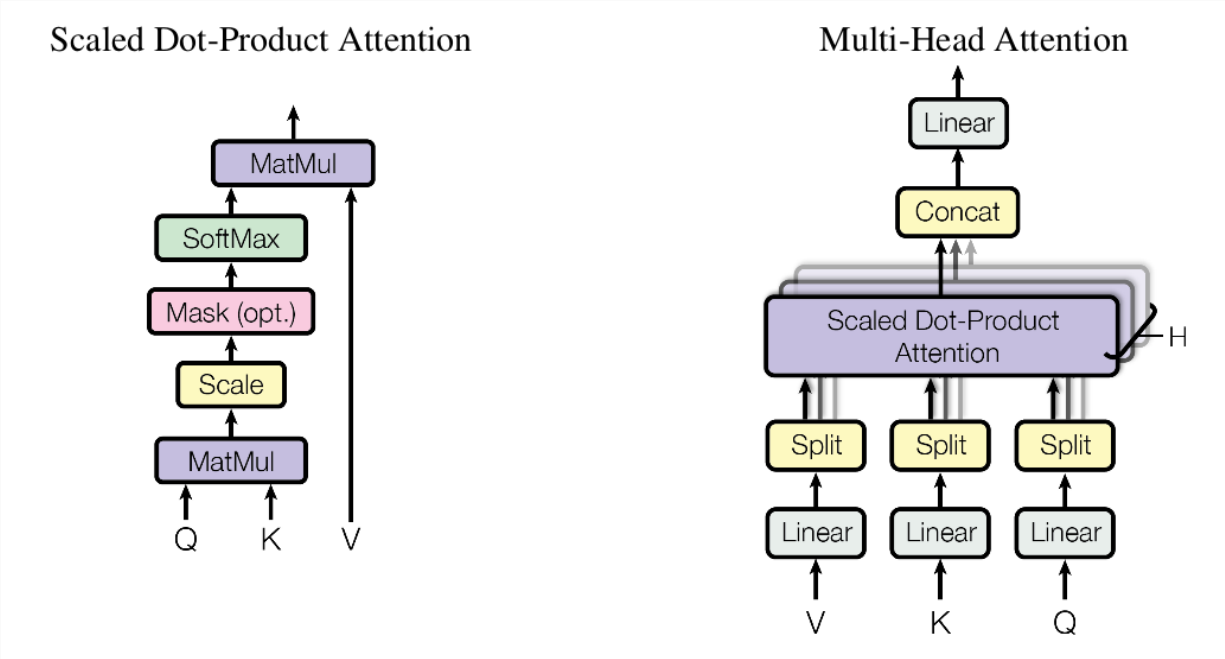
# ATTENTION IS ALL YOU NEED

- 12 июня 2017: «Attention is all you need» (Vaswani et al., Google)



# ATTENTION IS ALL YOU NEED

- Ничего, кроме внимания!
- Параллельные карты внимания, объединённые в матрицы:

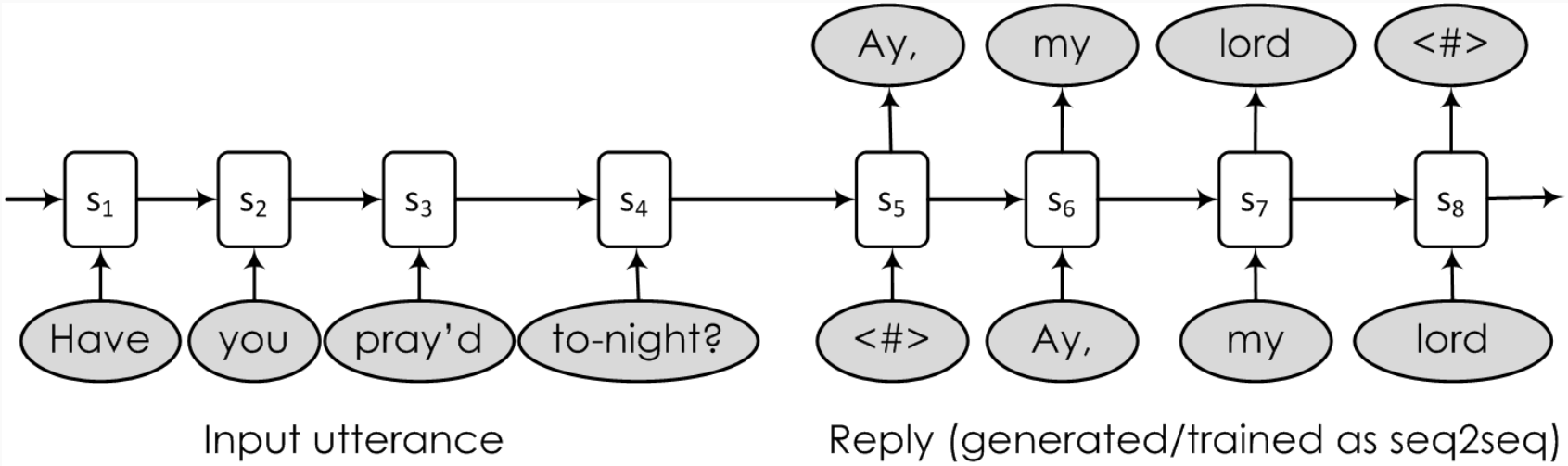



- Self-attention: каждая позиция encoder'а может внимательно посмотреть на каждую позицию предыдущего уровня.
- Результаты SMT лучше state of the art, а обучается в сто раз быстрее.

# ДИАЛОГОВЫЕ МОДЕЛИ

# ДИАЛОГОВЫЕ МОДЕЛИ

- Диалоговые модели пытаются предсказать развитие диалога или прямо активно разговаривают с человеком.
- (Vinyals, Le, 2015) используют *seq2seq* и encoder-decoder для диалога:
  - предыдущие предложения ABC подаём как контекст RNN;
  - следующее слово ответа WXYZ предсказываем из предыдущего и скрытого состояния.

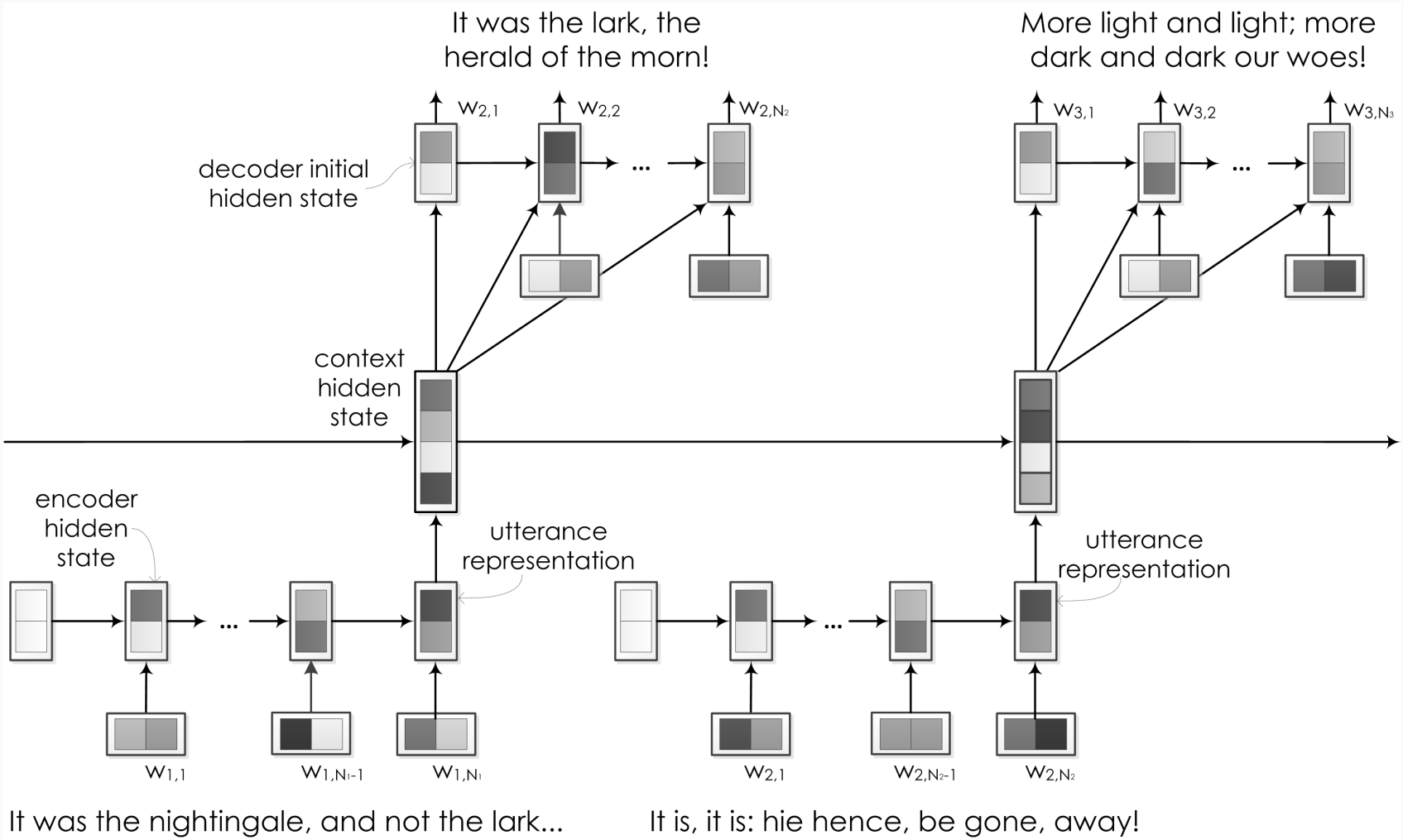



- Датасеты: общие (MovieSubtitles) и в конкретном домене (IT helpdesk).

# ДИАЛОГОВЫЕ МОДЕЛИ

- Hierarchical recurrent encoder decoder architecture (HRED); сначала для query suggestion в поиске (Sordoni et al. 2015), потом для диалоговых систем (Serban et al. 2015).
- Диалог – двухуровневая система: последовательность реплик, а каждая реплика – последовательность слов.
- HRED обучает:
  (1) *encoder* RNN, которая отображает каждую реплику в вектор;
  (2) *context* RNN, которая обрабатывает векторы предыдущих реплик и объединяет их в вектор текущего контекста;
  (3) *decoder* RNN, которая предсказывает слова следующей реплики при условии контекста.

- HRED (Serban et al. 2015):



# ДИАЛОГОВЫЕ МОДЕЛИ

- Некоторые недавние результаты и расширения:
  - (Su et al., 2016; Li et al., 2016a): обучение с подкреплением (DQN, online active reward learning) для улучшения порождения диалогов;
  - (Li et al., 2016b) добавляют *персонажей* (personas) при помощи скрытых переменных;
  - (Wen et al., 2016) используют *snapshot learning*, добавляя данные в виде неких событий, происходящих в выходном предложении (мы ещё хотим это сказать или уже сказали).
- В целом чатботы уже работают нормально, но до поддержания разговора общего назначения ещё очень далеко.

# ОТВЕЧАЕМ НА ВОПРОСЫ

# ОТВЕЧАЕМ НА ВОПРОСЫ

- Question answering (QA) – одна из сложнейших задач, тоже очень близка к пониманию.
- О каких вопросах речь:
  - легко найти датасеты из information retrieval;
  - фактические вопросы (тривия) из баз знаний;
  - классификация в случае вопросов на multiple choice (Quiz Bowl); тут хорошо работает отобразить вопрос и ответы в семантическое пространство и искать там ближайших соседей (Socher et al. 2014);
  - но это всё не совсем то.
- (Weston et al. 2015): датасет простых вопросов, которые «не требуют дополнительных знаний».
- Но требуют умения рассуждать и понимать семантику...

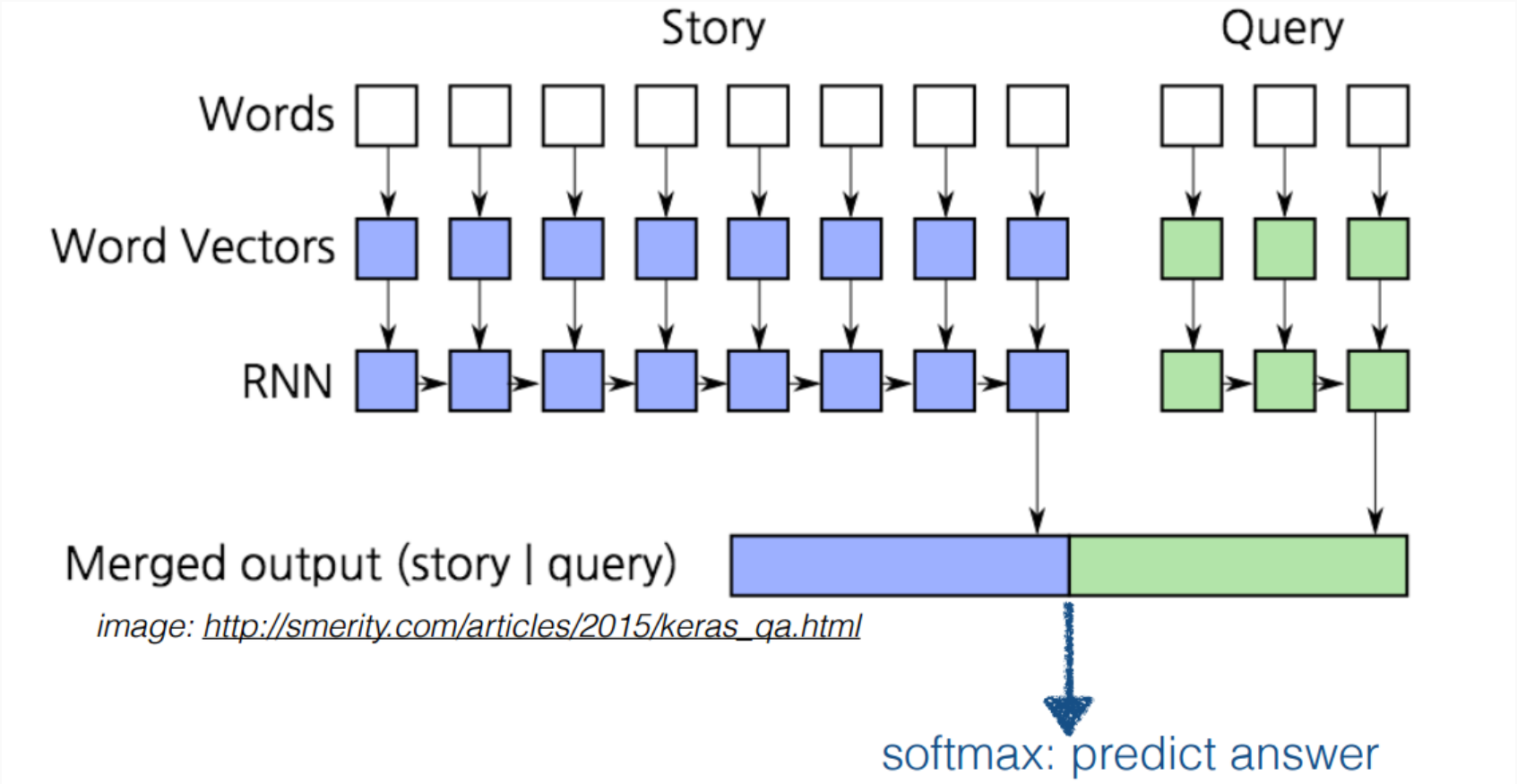
# ОТВЕЧАЕМ НА ВОПРОСЫ

- Примеры вопросов:

| | |
|---|---|
| **Task 1**: *Single Supporting Fact*<br>Mary went to the bathroom.<br>John moved to the hallway.<br>Mary travelled to the office.<br>Where is Mary?<br>**A**: *office* | **Task 4**: *Two Argument Relations*<br>The office is north of the bedroom.<br>The bedroom is north of the bathroom.<br>The kitchen is west of the garden.<br>What is north of the bedroom? **A**: *office*<br>What is the bedroom north of? **A**: *bathroom* |
| **Task 7**: *Counting*<br>Daniel picked up the football.<br>Daniel dropped the football.<br>Daniel got the milk.<br>Daniel took the apple.<br>How many objects is Daniel holding?<br>**A**: *two* | **Task 10**: *Indefinite Knowledge*<br>John is either in the classroom or the playground.<br>Sandra is in the garden.<br>Is John in the classroom?<br>**A**: *maybe*<br>Is John in the office?<br>**A**: *no* |
| **Task 15**: *Basic Deduction*<br>Sheep are afraid of wolves.<br>Cats are afraid of dogs.<br>Mice are afraid of cats.<br>Gertrude is a sheep.<br>What is Gertrude afraid of?<br>**A**: *wolves* | **Task 20**: *Agent's Motivations*<br>John is hungry.<br>John goes to the kitchen.<br>John grabbed the apple there.<br>Daniel is hungry.<br>Where does Daniel go? **A**: *kitchen*<br>Why did John go to the kitchen? **A**: *hungry* |

# ОТВЕЧАЕМ НА ВОПРОСЫ

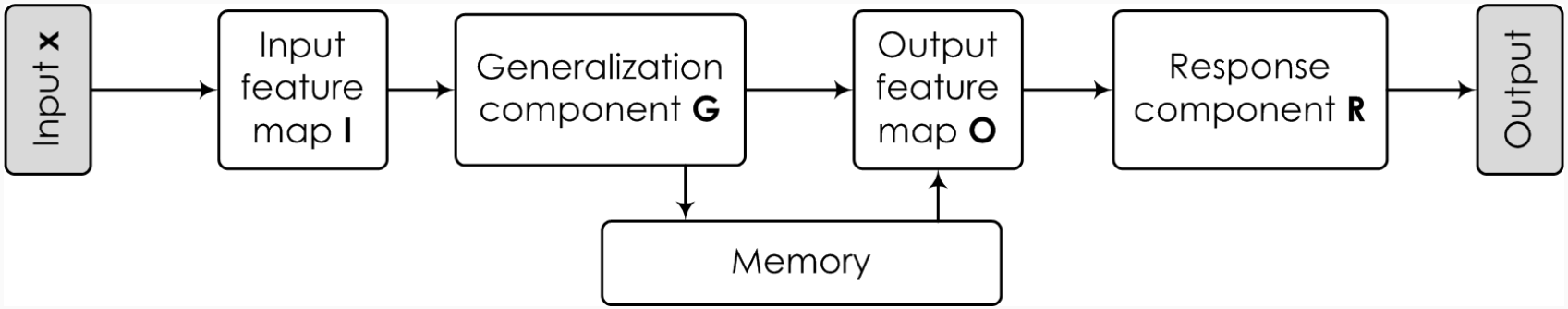
- Наивный подход – породить представления вопроса и ответа, совместить, отвечать:




- Проблема в том, что надо *запомнить* контекст на протяжении всего вопроса...

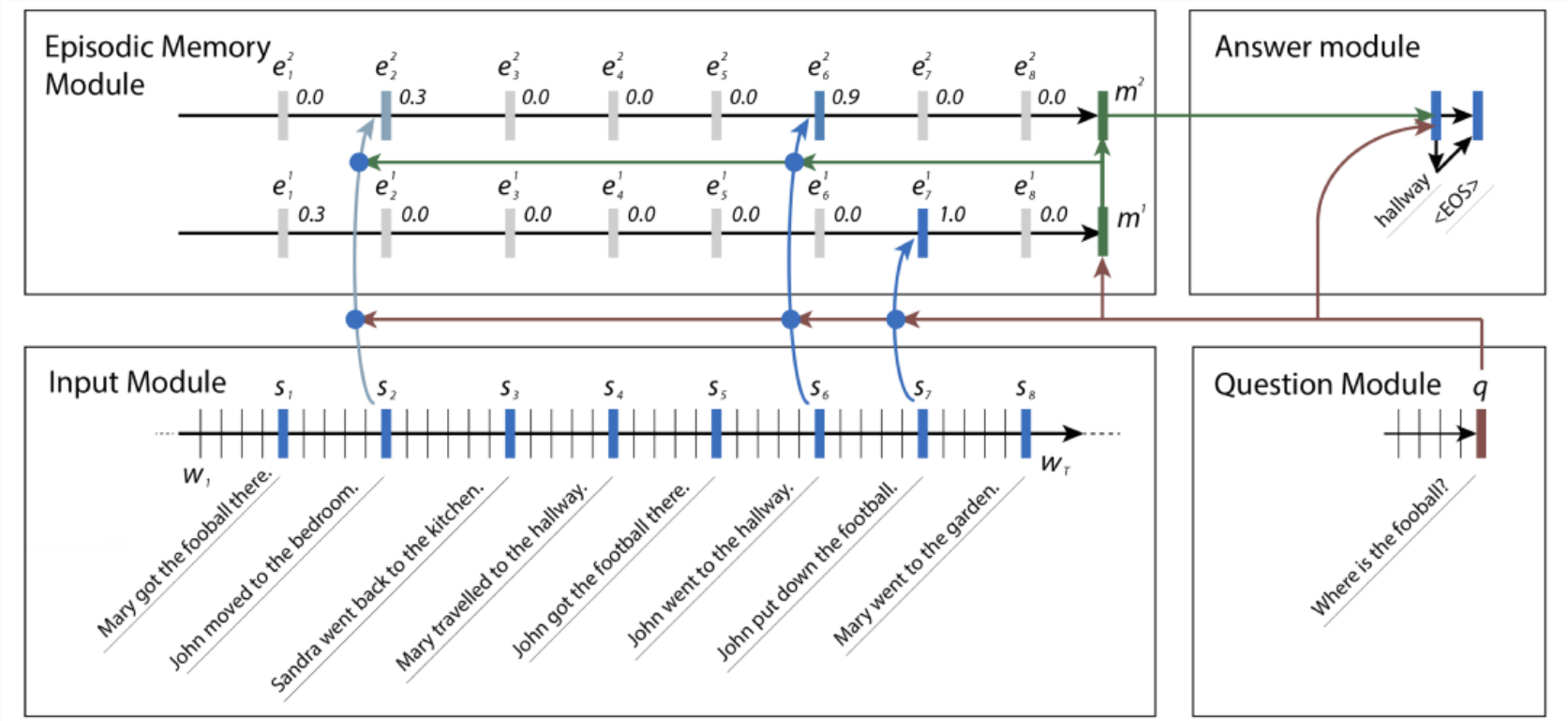
# ОТВЕЧАЕМ НА ВОПРОСЫ

- ...так что сейчас state of the art – это *memory networks* (Weston et al. 2014).
- Есть массив объектов (память) и обучаемые компоненты:
  - I (input feature map) преобразует вход во внутреннее представление;
  - G (generalization) обновляет память после нового входа;
  - O (output feature map) порождает выход по входу и состоянию памяти;
  - R (response) конвертирует выход O в нужный формат (генератор текста, например).



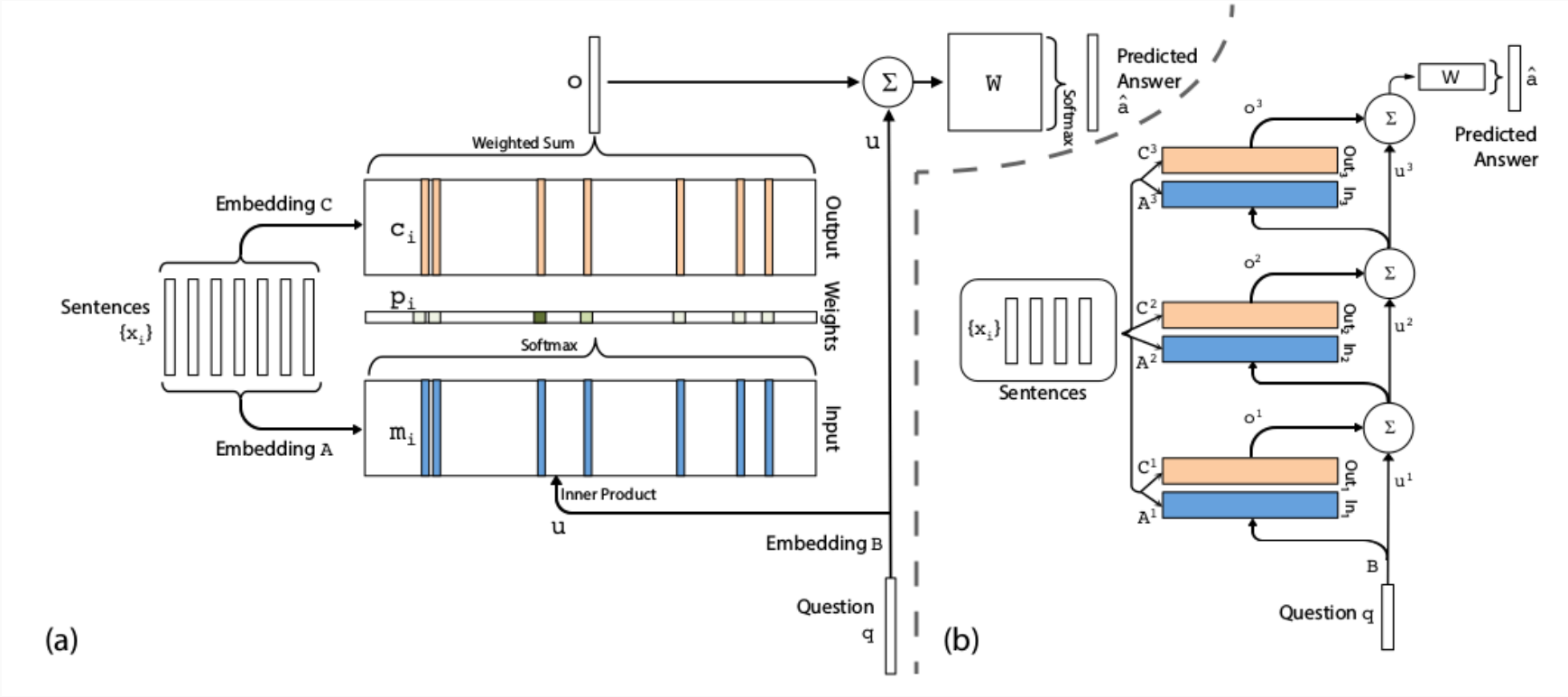
# ОТВЕЧАЕМ НА ВОПРОСЫ

- *Dynamic memory networks* (Kumar et al. 2015): эпизодическая память, которая выбирает, на какой части входа фокусироваться при помощи механизма внимания.



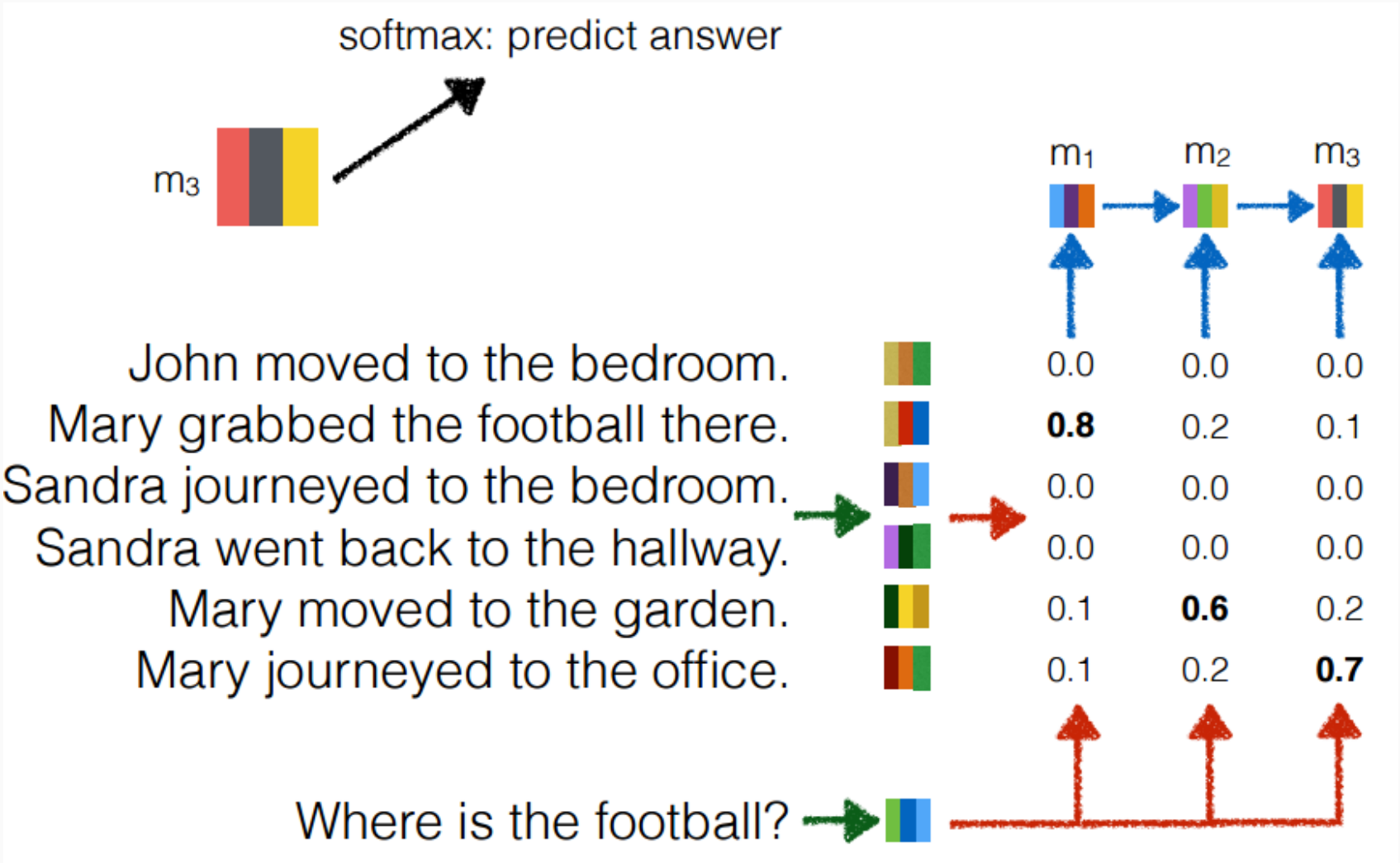
# ОТВЕЧАЕМ НА ВОПРОСЫ

- *End-to-end memory networks* (Sukhbaatar et al. 2015): непрерывная версия с несколькими шагами вычислений на каждый выход.



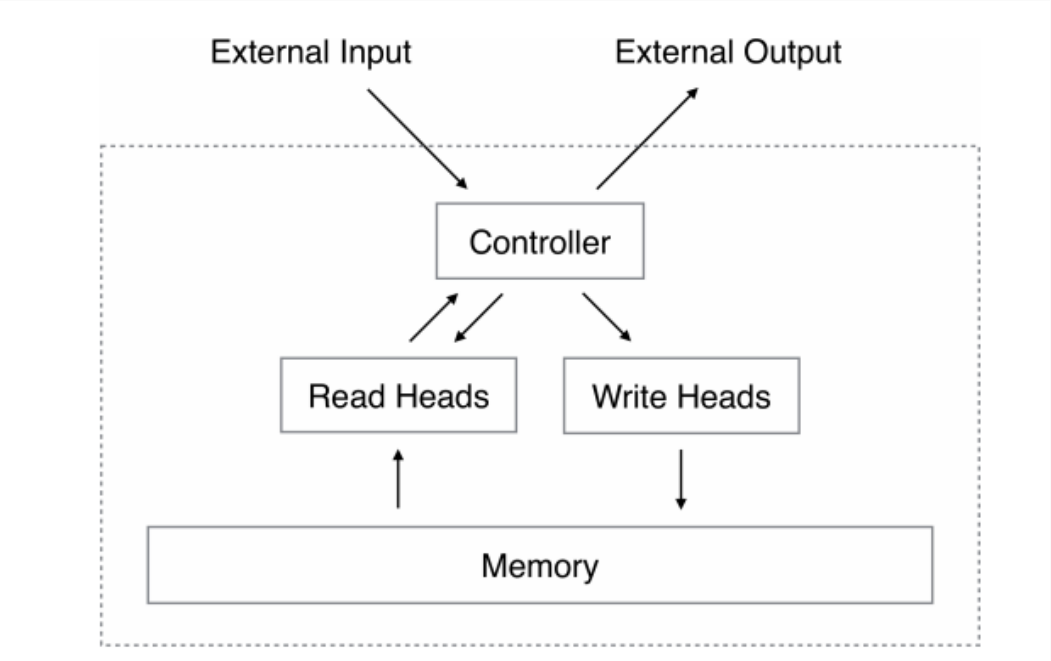
# ОТВЕЧАЕМ НА ВОПРОСЫ

- Должно примерно так работать:



# ОТВЕЧАЕМ НА ВОПРОСЫ

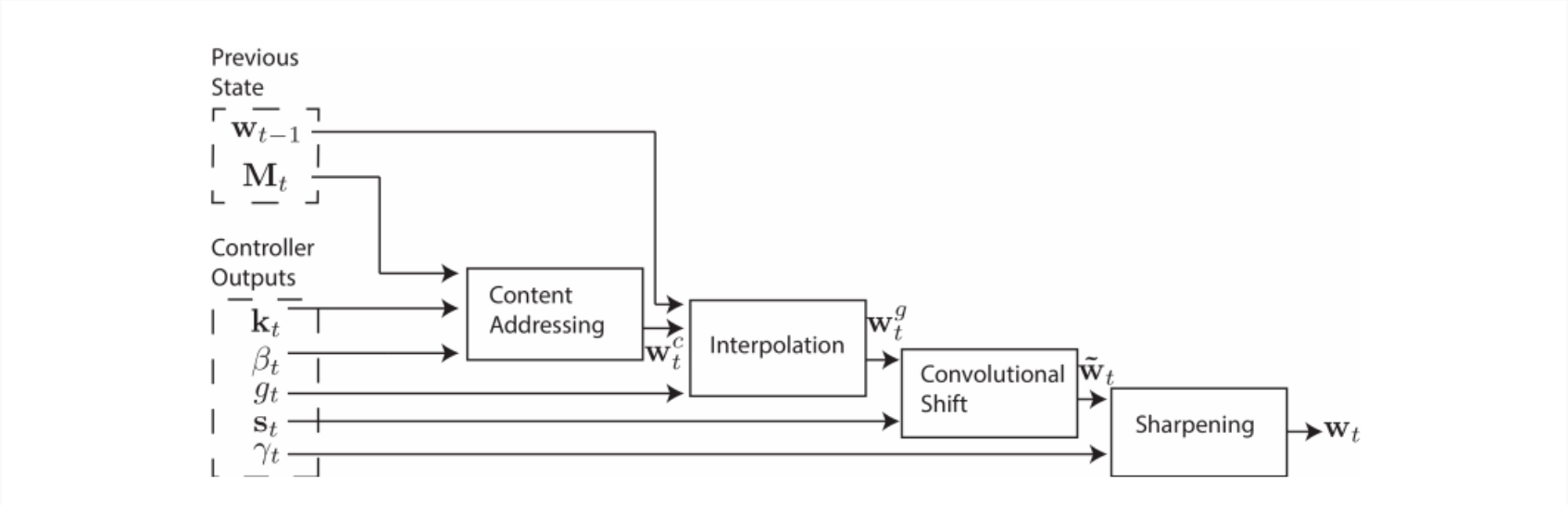
- Вообще идея того, чтобы обучать управляющие алгоритмы, может пойти далеко.
- Neural Turing Machines (Graves et al., 2014):



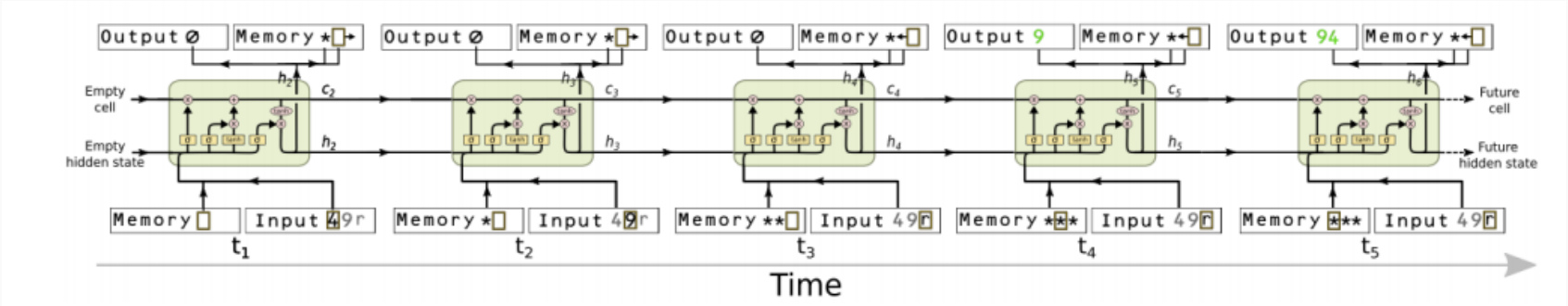

- Читают из памяти с весами вроде внимания, стирают память, пишут в неё.

# ОТВЕЧАЕМ НА ВОПРОСЫ

- Веса получаются механизмом адресации:




- Можно обучать с подкреплением (Zaremba, Sutskever, 2015):



# Отвечаем на вопросы

- Есть и другие расширения.
- Пока не до конца понятно, как связать memory networks с базами знаний, собственно.
- Впереди много интересного. Например...

# Что? Где? Когда?

- «Что? Где? Когда?» иногда выглядит так...

# ЧТО? ГДЕ? КОГДА?

- …но обычно примерно так:

# что? где? когда?

- Команды из $\leq 6$ игроков отвечают на вопросы.
- db.chgk.info – база около $300$K вопросов.
- Некоторые из «Своей игры», в которой вопросы обычно гораздо интереснее Jeopardy:
  - *Логотипы*

    **ЕГО** логотип — сочетание рун «беркана» и «хаглаз», инициалов конунга Харальда.
  - *Сокращения*

    Учитывая **ЕГО** большую роль в создании первых танков, эти танки часто шутливо именовали ватерклозетами.
  - *Лошади*

    **ЕГО** попытка приударить за галисийскими кобылицами окончилась печальным образом.

# что? где? когда?

- А большинство – вопросы «Что? Где? Когда?», ещё сложнее:
  - Однажды Ричард Фейнман объяснял своей девушке, почему Декарт говорил глупости, когда доказывал существование Бога. Не желая ввязываться в спор, девушка сказала, что на любой предмет, видимо, можно посмотреть с разных сторон. Ответьте двумя словами: что сделал Фейнман в ответ на это?
  - Вторая соответствовала богохульству, третья — подделке денег или документов, четвёртая — пьянству. А чему соответствовала первая?
  - На самом деле чехословацкая линейка аудиотехники и радиокомпонентов получила название потому, что это слаботочная техника. Напишите это название.
  - Одна из конструкций математической логики рассматривает бессмертного агента, который в каждый момент времени должен либо совершить действие, либо формально доказать, что он совершит это действие в один из последующих моментов времени. Название этой конструкции состоит из двух слов, начинающихся на одну и ту же букву. Напишите это название.
- Отличный и пока нереально сложный датасет. Когда? :)

# СПАСИБО!

Спасибо за внимание!