

Автоматическая обработка текстов



Языковые модели

Лекция 5

Емельянов А. А.
login-const@mail.ru

- **Цель языкового моделирования** – это оценка распределение вероятностей лингвистических единиц:
 - Символов
 - Слов
 - Других токенов последовательности

Где используются языковые модели

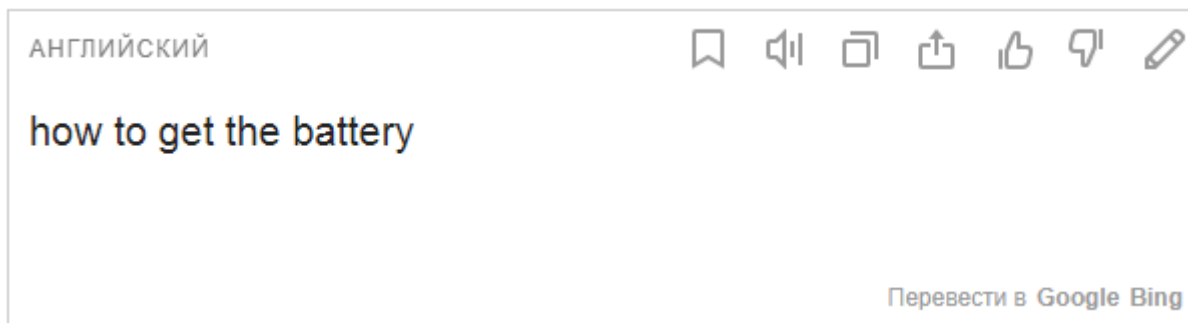
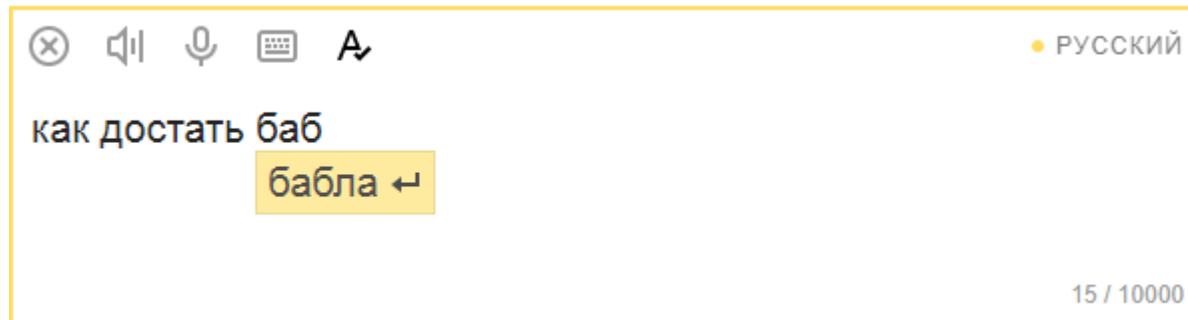
как достать ба  

как достать **батарею** из часов
как достать **батарею** из ноутбука
как достать **батарейки** из газовой колонки
как достать **батарею** из айфона
как достать **батарею** из ключа бмв
как достать **банку** из кипятка
как достать **батарею** из ноутбука асус
как достать **барабан** из стиральной машины индезит
как достать **батарею** из ноутбука асер
как достать **бабла**

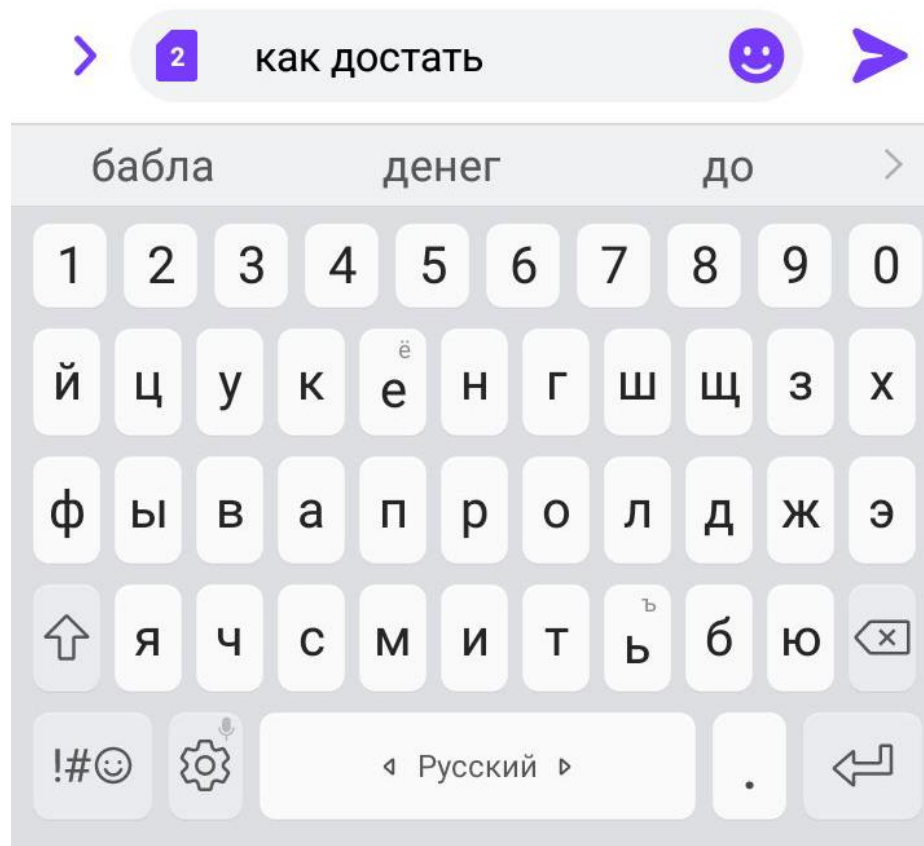
Поиск в Google Мне повезёт!

Пожаловаться на неприемлемые подсказки

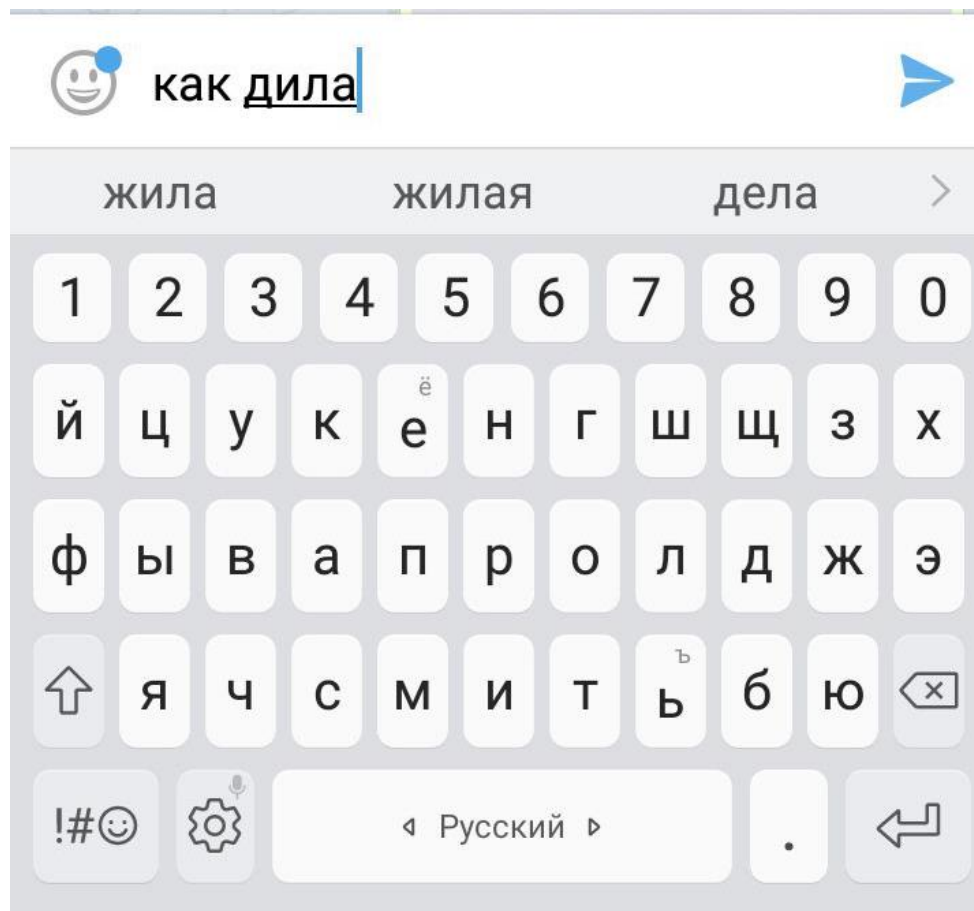
Где используются языковые модели



Где используются языковые модели



Где используются языковые модели



Мы используем языковые модели каждый день

- Мы можем выбрать один вариант между одинаково звучащими (похожими) фразами:

Я хочу назвать моего кота наполеон.

Я хочу назвать моего кота на поле он.



Я хочу назвать моего кота Наполеон.

- Для автоматических моделей в решении поможет **вероятность предложения**.

Вероятность предложений: интуиция

- «Вероятность предложения» = насколько это возможно в естественном языке.
- Рассматривается только конкретный язык:

$P(\text{Кот лежит на диване}) > P(\text{На диване кот лежит})$

$P(\text{Красивая девочка играла в мяч}) > P(\text{В мяч играла девочка красивая})$

- Очень сложно знать настоящую вероятность последовательности токенов.
- Но мы можем использовать **языковую модель** для приближения этой вероятности.
- Как все модели, языковые модели в одних случаях «ведут себя хорошо» в других «плохо».

- Языковые модели можно разделить на два типа:
 - **Count-based Models (Statistical)** – статистические языковые модели.
 - **Neural Language Models** – языковые модели на основе нейронных сетей.

N-gram LM: count-based LM

- Как оценить вероятность встретить предложение $s = (x^{(1)}, x^{(2)}, \dots, x^{(n)})$ в рамках конкретного языка (например русского)?
- Предположим у нас есть обучающие данные: *большой* текст (корпус C) на русском языке.
- И мы разбили его на предложения.
- Мы хотим оценить вероятность s , используя имеющиеся данные.

N-gram LM: count-based LM

- Как оценить вероятность встретить предложение $s = (x^{(1)}, x^{(2)}, \dots, x^{(n)})$ в рамках конкретного языка (например русского)?
- Предположим у нас есть обучающие данные: *большой* текст (корпус C) на русском языке.
- И мы разбили его на предложения.
- Мы хотим оценить вероятность s , используя имеющиеся данные.

$$MLE: P(s) = \frac{Count(s \in C)}{|C|}$$

N-gram LM: count-based LM

- Как оценить вероятность встретить предложение $s = (x^{(1)}, x^{(2)}, \dots, x^{(n)})$ в рамках конкретного языка (например русского)?
- Предположим у нас есть обучающие данные: *большой* текст (корпус C) на русском языке.
- И мы разбили его на предложения.
- Мы хотим оценить вероятность s , используя имеющиеся данные.

$$MLE: P(s) = \frac{Count(s \in C)}{|C|}$$

- Что делать, если предложение s не встретилось ни разу в данных?

N-gram LM: count-based LM

- Что делать, если предложение s не встретилось ни разу в данных?

s_1 = «Длина тела кархародонтозавра достигала 12 метров.»

VS

s_2 = «достигала 12 метров.»

- Первое предложение s_1 более осмысленно чем второе – s_2 .
- Но вероятность предложения s_1 равна $P(s_1) = 0$, а вероятность второго предложения s_2 может быть ненулевой $P(s_2) > 0$!

N-gram LM: count-based LM

- Что делать, если предложение s не встретилось ни разу в данных?

s_1 = «Длина тела кархародонтозавра достигала 12 метров.»

VS

s_2 = «достигала 12 метров.»

- Первое предложение s_1 более осмысленно чем второе – s_2 .
- Но вероятность предложения s_1 равна $P(s_1) = 0$, а вероятность второго предложения s_2 может быть ненулевой $P(s_2) > 0$!



- Предыдущий подход *MLE* **не работает на полных предложениях.**

N-gram LM: count-based LM

- Что делать, если предложение s не встретилось ни разу в данных?
- Идея:
 - Приблизим вероятность $P(s)$ скомбинировав вероятности меньших частей предложения, которые встречаются более часто.
- Решение:
 - Получим N-gram Language Model

N-gram LM: введение

- Хотим оценить вероятность последовательности слов (токенов): $P(s = (x^{(1)}, x^{(2)}, \dots, x^{(n)}))$.

- Пример: «Я хочу назвать кота Наполеон»

- Это совместная вероятность встретить слова (токены) $x^{(i)}$ в предложении s .

- Совместная вероятность:
 $P(X, Y) = P(Y|X)P(X)$.

- Тогда

$$\begin{aligned} P(\text{Я хочу назвать кота Наполеон}) &= \\ &= P(\text{Наполеон}|\text{Я хочу назвать кота}) \times P(\text{Я хочу назвать кота}) = \\ &= P(\text{Наполеон}|\text{Я хочу назвать кота}) \times P(\text{кота}|\text{Я хочу назвать}) \times \\ &\quad P(\text{назвать}|\text{Я хочу}) \times P(\text{хочу}|\text{Я}) \times P(\text{Я}) \end{aligned}$$



N-gram LM: введение

- Хотим оценить вероятность последовательности слов (токенов): $P(s = (x^{(1)}, x^{(2)}, \dots, x^{(n)}))$.

- Пример: «Я хочу назвать кота Наполеон»

- Это совместная вероятность встретить слова (токены) $x^{(i)}$ в предложении s .

- Совместная вероятность:
 $P(X, Y) = P(Y|X)P(X)$.

- Тогда

$$\begin{aligned} P(\text{Я хочу назвать кота Наполеон}) &= \\ &= P(\text{Наполеон} | \text{Я хочу назвать кота}) \times P(\text{Я хочу назвать кота}) = \\ &= P(\text{Наполеон} | \text{Я хочу назвать кота}) \times P(\text{кота} | \text{Я хочу назвать}) \times \\ &\quad P(\text{назвать} | \text{Я хочу}) \times P(\text{хочу} | \text{Я}) \times P(\text{Я}) \end{aligned}$$

- Какая проблема остается?



N-gram LM: введение

- Какая проблема остается?

- Chain rule дает нам:

$$P(x^{(1)}, x^{(2)}, \dots, x^{(n)}) = \prod_i^n P(x^{(i)} | x^{(1)}, \dots, x^{(i-1)})$$

- Но много условных вероятностей все равно нулевые!
- Если мы хотим получить вероятность:

$P(\text{Длина тела кархародонтозавра достигала 12 метров})$

- Необходимо иметь вероятность:

$P(\text{метров} | \text{Длина тела кархародонтозавра достигала 12})$

N-gram LM: введение

- Сделаем **предположение о независимости**: вероятность слова (токена) зависит только от фиксированного числа предыдущих слов (**истории**).
- **Марковское предположение**:

$$P(x^{(i)} | x^{(1)}, \dots, x^{(i-1)}) = P(x^{(i)} | x^{(i-n+1)}, \dots, x^{(i-1)})$$

- Триграммная модель (trigram model):

$$P(x^{(i)} | x^{(1)}, \dots, x^{(i-1)}) \approx P(x^{(i)} | x^{(i-2)}, x^{(i-1)})$$

- Биграммная модель (bigram model):

$$P(x^{(i)} | x^{(1)}, \dots, x^{(i-1)}) \approx P(x^{(i)} | x^{(i-1)})$$

- Униграммная модель (unigram model):

$$P(x^{(i)} | x^{(1)}, \dots, x^{(i-1)}) \approx P(x^{(i)})$$

N-gram LM

$$P(x^{(i)} | x^{(1)}, \dots, x^{(i-1)}) = P(x^{(i)} | \overbrace{x^{(i-n+1)}, \dots, x^{(i-1)}}^{n-1 \text{ слов}}) =$$

наше предположение

вероятность n-gram

$$= \frac{P(x^{(i-n+1)}, \dots, x^{(i)})}{P(x^{(i-n+1)}, \dots, x^{(i-1)})} =$$

определение условной вероятности

вероятность (n-1)-gram

- **Вопрос:** как получить вероятность n-gram и (n-1)-gram?

N-gram LM

$$P(x^{(i)} | x^{(1)}, \dots, x^{(i-1)}) = P(x^{(i)} | \overbrace{x^{(i-n+1)}, \dots, x^{(i-1)}}^{n-1 \text{ слов}}) =$$

наше предположение

$$\overset{\text{вероятность n-gram}}{\longrightarrow} \frac{P(x^{(i-n+1)}, \dots, x^{(i)})}{P(x^{(i-n+1)}, \dots, x^{(i-1)})} =$$

определение условной вероятности

\swarrow
вероятность (n-1)-gram

- **Вопрос:** как получить вероятность n-gram и (n-1)-gram?
- **Ответ:** Вычислим по большому обучающему тексту (корпусу), который у нас есть.

$$\approx \frac{\text{Count}(x^{(i-n+1)}, \dots, x^{(i)})}{\text{Count}(x^{(i-n+1)}, \dots, x^{(i-1)})}$$

статистическое приближение

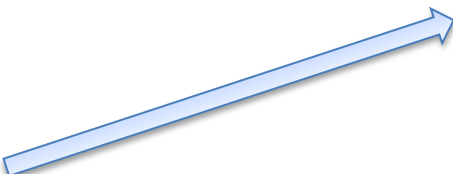
Пример: 4-gram model

- ~~Котик~~ очень тихо спал на _____
не участвует условия на этом



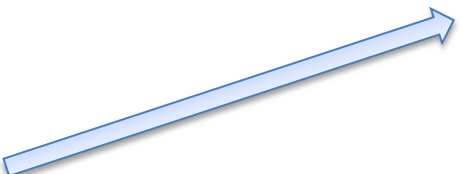
$$P(x^{(i)} | \text{Котик очень тихо спал на}) \approx \frac{\text{Count}(\text{тихо спал на } x^{(i)})}{\text{Count}(\text{тихо спал на})}$$

Проблемы с n-gram models

$$P(x^{(i)} | \text{Котик очень тихо спал на}) \approx \frac{\text{Count}(\text{тихо спал на } x^{(i)})}{\text{Count}(\text{тихо спал на})}$$


Проблема: что если «тихо спал на» не встретилось в данных. Тогда нет возможности вычислить вероятность $x^{(i)}$!

Проблемы с n-gram models

$$P(x^{(i)} | \text{Котик очень тихо спал на}) \approx \frac{\text{Count}(\text{тихо спал на } x^{(i)})}{\text{Count}(\text{тихо спал на})}$$


The diagram consists of a blue arrow pointing from the denominator of the formula, 'Count(тихо спал на)', down to the 'Проблема' box. Another blue arrow points from the 'Проблема' box to the '(Частичное) решение' box.

Проблема: что если «тихо спал на» не встретилось в данных. Тогда нет возможности вычислить вероятность $x^{(i)}$!

(Частичное) решение:
backoff сглаживание.

- Иногда помогает использование меньшего контекста

- Backoff:**

- Используем trigram, если есть в выборке
- иначе bigram
- иначе unigram



Если нет «тихо спал на», пробуем «спал на».

$$P(x^{(i)} | \text{тихо спал на}) \approx P(x^{(i)} | \text{спал на})$$



Если нет «спал на», пробуем «на».

$$P(x^{(i)} | \text{спал на}) \approx P(x^{(i)} | \text{на})$$



Если нет «на», пробуем $x^{(i)}$.

$$P(x^{(i)} | \text{на}) \approx P(x^{(i)})$$

Линейная интерполяция

- Интерполяция смешивает unigram, bigram, trigram...

$$\begin{aligned}\hat{P}(x^{(i)} | x^{(i-3)}, x^{(i-2)}, x^{(i-1)}) \approx & \lambda_3 P(x^{(i)} | x^{(i-3)}, x^{(i-2)}, x^{(i-1)}) + \\ & \lambda_2 P(x^{(i)} | x^{(i-2)}, x^{(i-1)}) + \\ & \lambda_1 P(x^{(i)} | x^{(i-1)}) + \\ & \lambda_0 P(x^{(i)})\end{aligned}$$
$$\sum_{i=0}^{n-1} \lambda_i = 1$$

- Вопрос:** как выбрать λ_i ?

Линейная интерполяция

- Интерполяция смешивает unigram, bigram, trigram...

$$\begin{aligned}\hat{P}(x^{(i)} | x^{(i-3)}, x^{(i-2)}, x^{(i-1)}) \approx & \lambda_3 P(x^{(i)} | x^{(i-3)}, x^{(i-2)}, x^{(i-1)}) + \\ & \lambda_2 P(x^{(i)} | x^{(i-2)}, x^{(i-1)}) + \\ & \lambda_1 P(x^{(i)} | x^{(i-1)}) + \\ & \lambda_0 P(x^{(i)})\end{aligned}$$
$$\sum_{i=0}^{n-1} \lambda_i = 1$$

- Вопрос:** как выбрать λ_i ?
- Ответ:** использовать валидационный набор данных.

Проблемы с n-gram models

Проблема: что если «тихо спал на $x^{(i)}$ » не встретилось в данных. Тогда вероятность $x^{(i)}$ равна 0!

(Частичное) решение: различные сглаживания (add-one, Kneser-Neu, др.)

$$P(x^{(i)} | \text{Котик очень тихо спал на}) \approx \frac{\text{Count}(\text{тихо спал на } x^{(i)})}{\text{Count}(\text{тихо спал на})}$$

Проблема: что если «тихо спал на» не встретилось в данных. Тогда нет возможности вычислить вероятность $x^{(i)}$!

(Частичное) решение: backoff сглаживание.

Сглаживание Лапласа (add-one)

- Предположим, что каждое слово (n-gram) встретилось хотя бы 1 раз.
- Добавим 1 к числителю и знаменателю.
- Если 1 – слишком грубая оценка, то добавим δ для каждого слова $x^{(i)} \in V$.

$$\hat{P}(x^{(i)} | x^{(i-n+1)}, \dots, x^{(i)}) = \frac{\delta + P(x^{(i-n+1)}, \dots, x^{(i)})}{\delta|V| + P(x^{(i-n+1)}, \dots, x^{(i-1)})}$$

Kneser-Ney сглаживание

- Вводится умный способ построения модели backoff.
- Рассматривает частоту unigram относительно возможных слов, предшествующих неизвестному слову.
- Метод, прежде всего раньше вычислял распределение вероятности n -gramm в документе, основанном на их историях.
- Этот подход считается одинаково эффективным и для более высоких и для n -граммов более низкоуровневых.
- Общим примером, который иллюстрирует понятие проблемы, которую решает этот метод, является частота биграммы «Сан-Франциско». Если это будет несколько раз появляться в тренировочном корпусе, то частота unigram «Франсиско» также будет высока. Если смотреть только по unigram частоте, чтобы предсказать частоты n -gram приводит к перекошенным результатам; однако, сглаживание Kneser–Ney исправляет это, рассматривая частоту unigram относительно возможных слов, предшествующих ему.

Как оценивать языковые модели

- Внутреннее оценивание:

- Cross-entropy:

$$H_M(w_1 w_2 \dots w_n) = -\frac{1}{n} \cdot \log P_M(w_1 w_2 \dots w_n)$$

- показывает насколько хорошо модель умеет предсказывать следующее слово.

- Perplexity (указывается в статьях):

$$\text{perplexity} = 2^{\text{cross-entropy}}$$

- Внешнее оценивание:

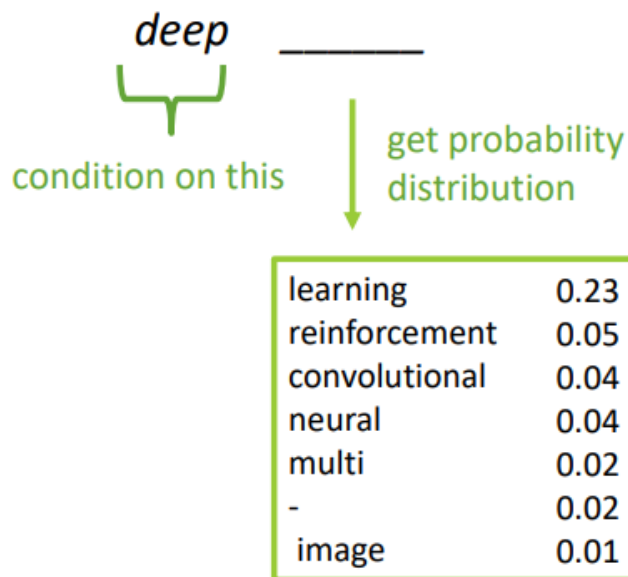
- Конкретная задача: заменяем одну языковую модель на другую и смотрим метрику качества данной задачи.

- Языковые модели можно разделить на два типа:
 - **Count-based Models (Statistical)** – статистические языковые модели.
 - Марковское предположение порядка n ;
 - Аппроксимация вероятностей n -gram (counting and smoothing).
 - **Neural Language Models** – языковые модели на основе нейронных сетей.

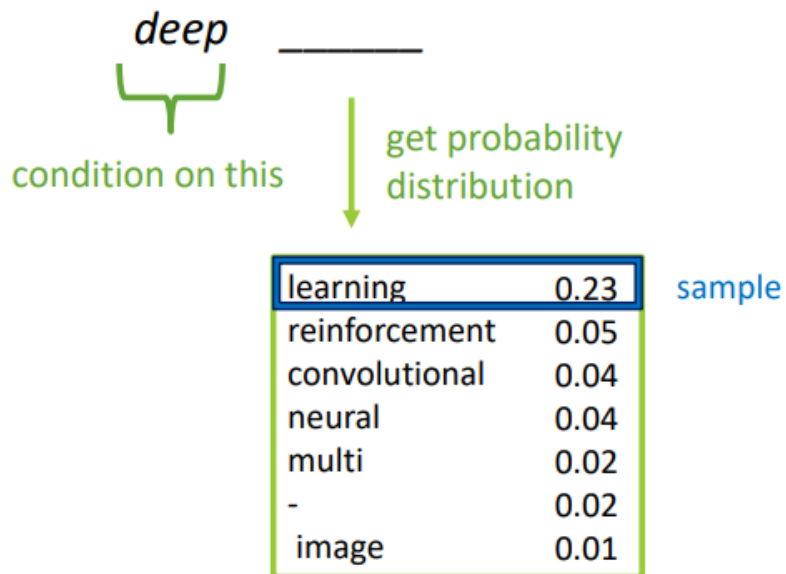
How to generate text using N-gram LM?

deep _____
└─┘
condition on this

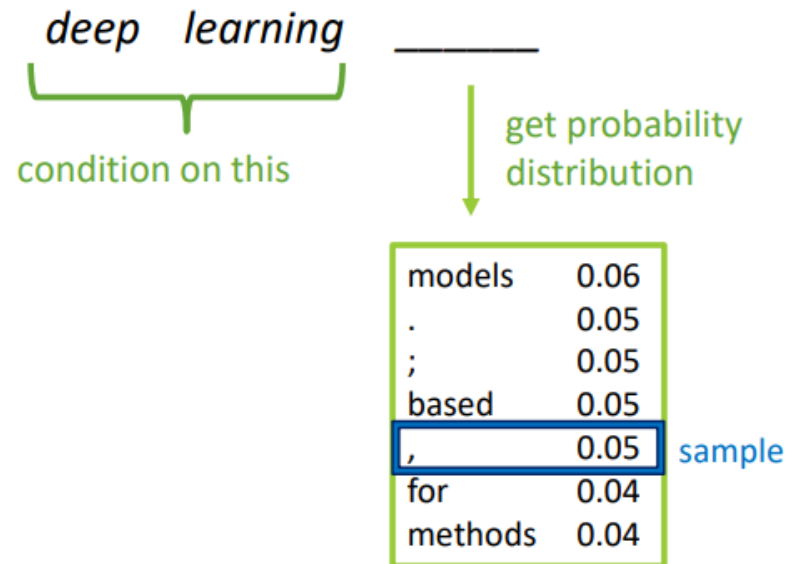
How to generate text using N-gram LM?



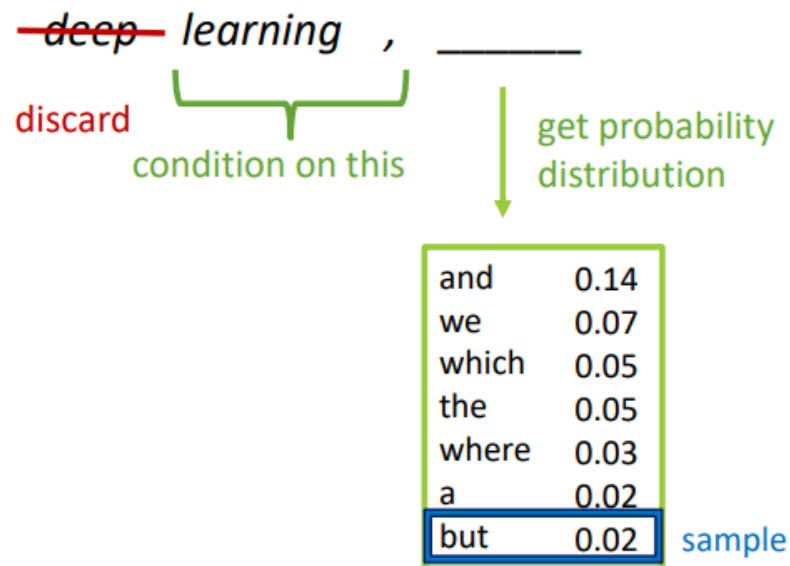
How to generate text using N-gram LM?



How to generate text using N-gram LM?



How to generate text using N-gram LM?



How to generate text using N-gram LM?

deep learning , but also is central to human performance . however , using structural similarity index measure than other partitioned sampling schemes , while making the approach with empirical data has the effect of phonetics has received little attention within the context of information on ...

Что не так с данным текстом?

How to generate text using N-gram LM?

deep learning , but also is central to human performance . however , using structural similarity index measure than other partitioned sampling schemes , while making the approach with empirical data has the effect of phonetics has received little attention within the context of information on ...

Что не так с данным текстом?

Он полностью бессмысленный!

Neural Language Models (NLM)

- Нейронные языковые модели с фиксированным окном

- Итоговое распределение:

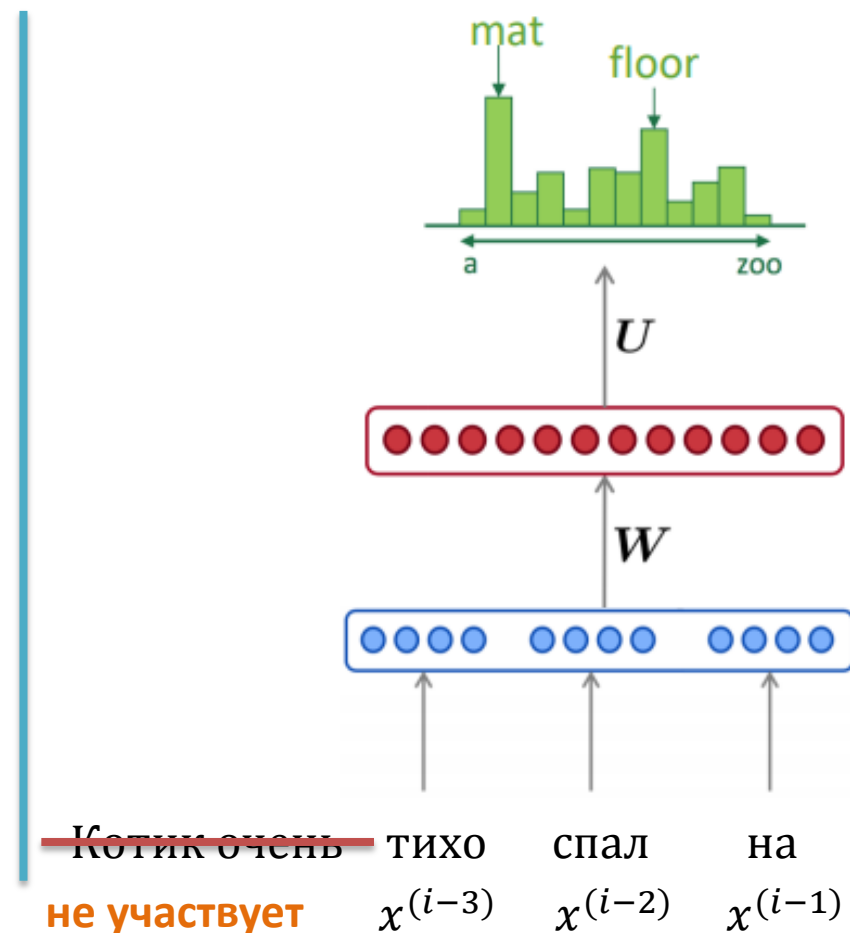
$$\hat{y} = \text{softmax}(Uh + b_2) \in R^{|V|}$$

- Скрытый слой (или другая feed-forward NN)

$$h = f(Wx + b_1)$$

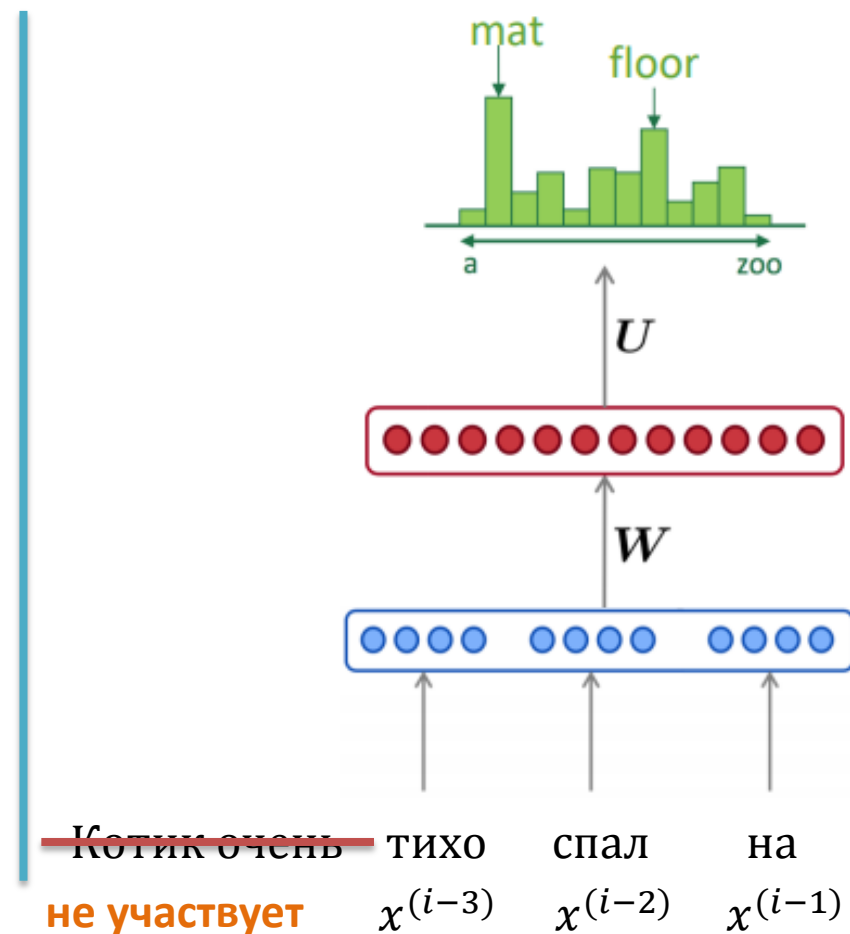
- Конкатенация эмбеддингов
 $x = (x^{(i-3)}, x^{(i-2)}, x^{(i-1)})$

- Эмбеддинги слов



Neural Language Models (NLM)

- Нейронные языковые модели с фиксированным окном
- Какие **улучшения** в сравнении с N-gram LM:

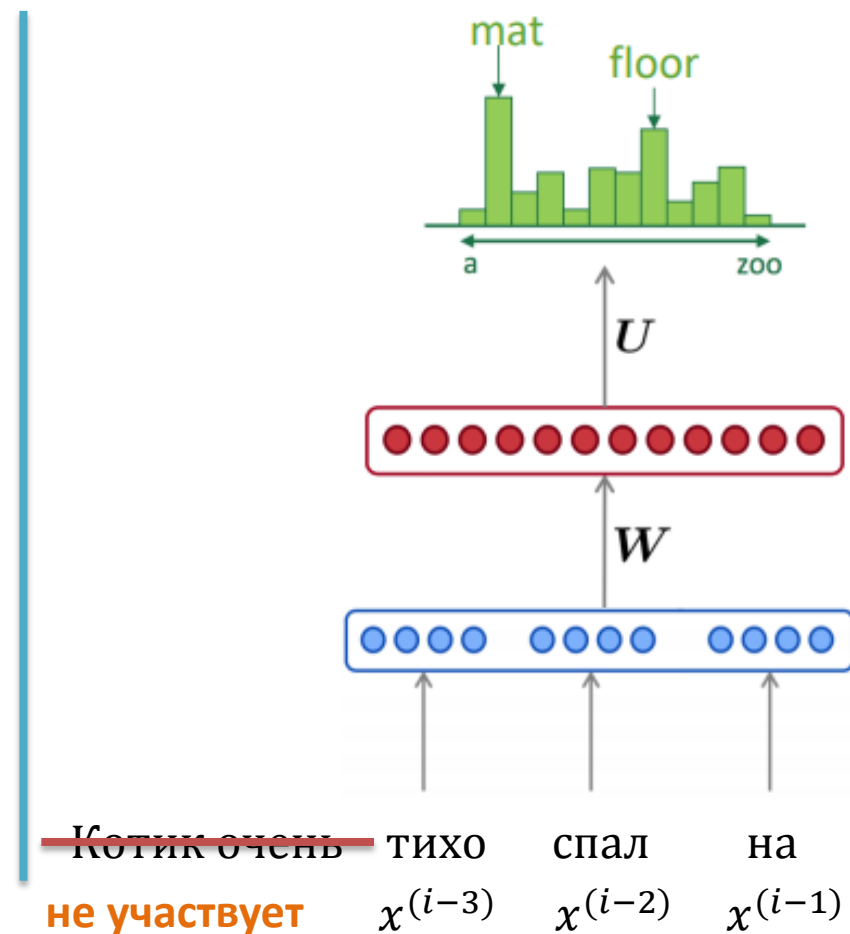


Neural Language Models (NLM)

- Нейронные языковые модели с фиксированным окном

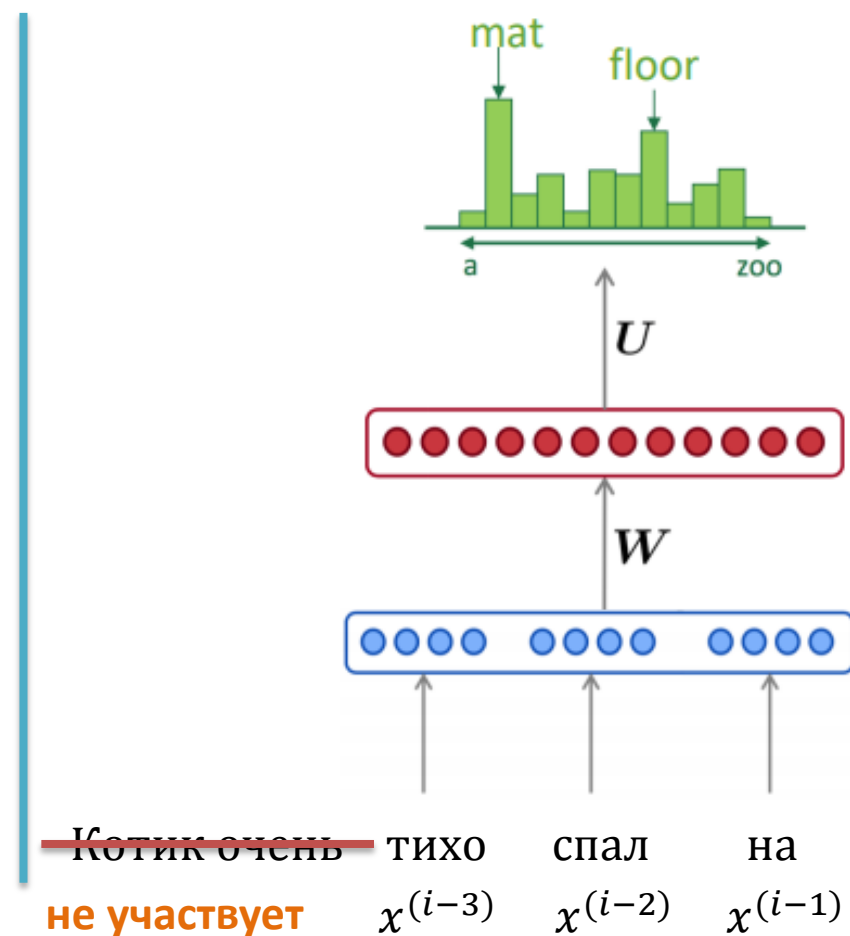
- Какие **улучшения** в сравнении с N-gram LM:

- Нет проблемы разреженности
- Размер модели $O(V)$ (вместо $O(\exp(V))$)



Neural Language Models (NLM)

- Нейронные языковые модели с фиксированным окном
- Какие **улучшения** в сравнении с N-gram LM:
 - Нет проблемы разреженности
 - Размер модели $O(V)$ (вместо $O(\exp(V))$)
- Какие **проблемы** остались:



Neural Language Models (NLM)

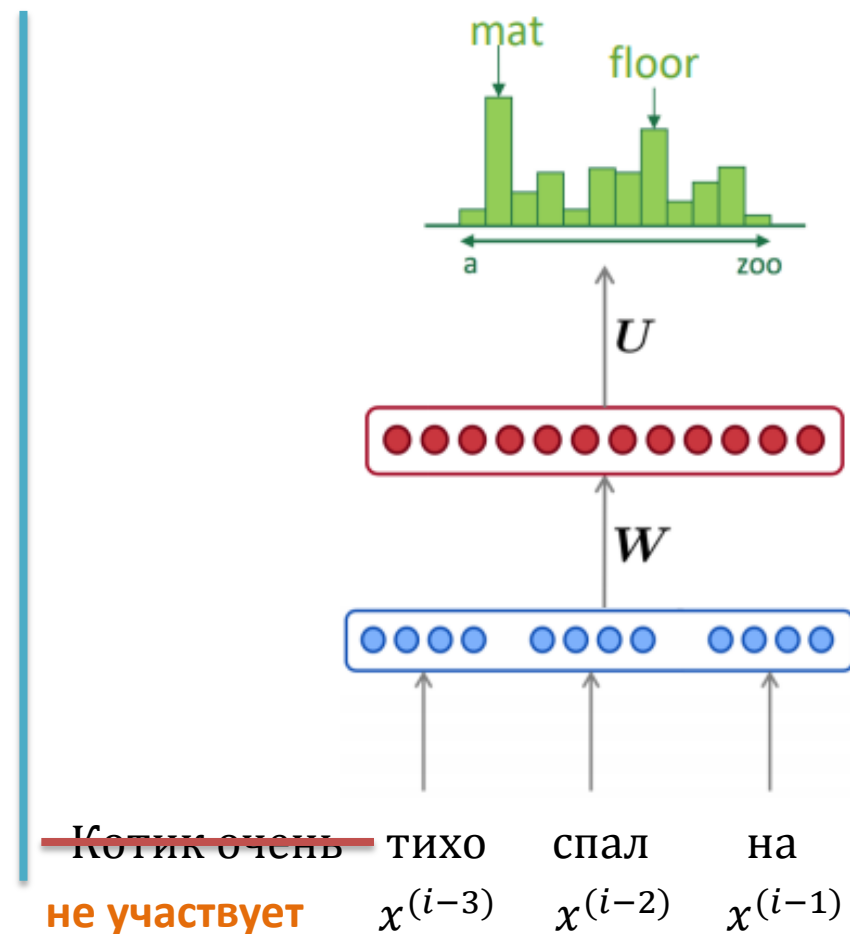
- Нейронные языковые модели с фиксированным окном

- Какие **улучшения** в сравнении с N-gram LM:

- Нет проблемы разреженности
- Размер модели $O(V)$ (вместо $O(\exp(V))$)

- Какие **проблемы** остались:

- Фиксированный размер окна – очень мал, нельзя поменять
- Фиксированный порядок слов
- Можно использовать веса только внутри окна



Neural Language Models (NLM)

- Рекуррентные языковые модели (RNN LM)

- Итоговое распределение:

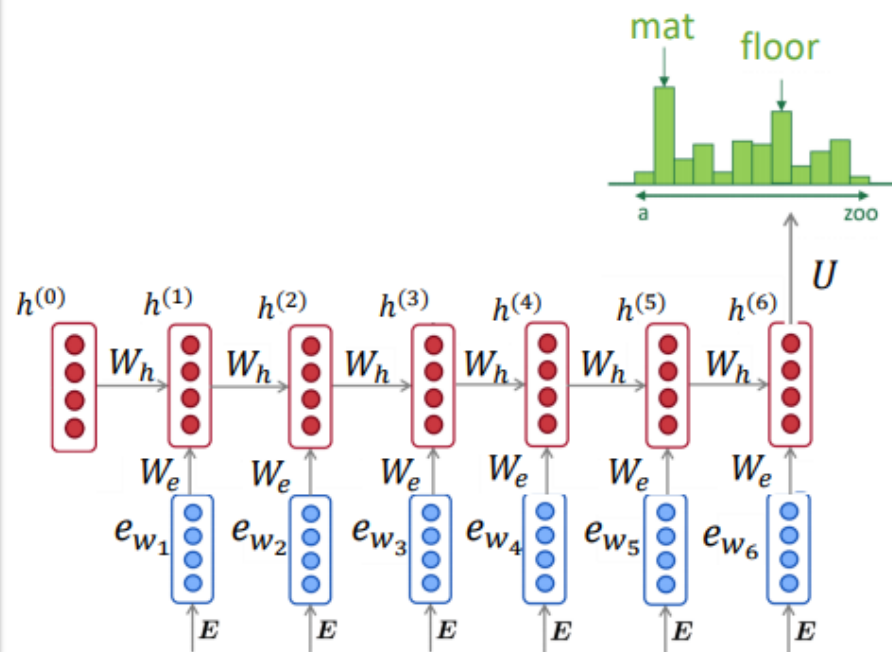
$$\hat{y}_t = \text{softmax}(Uh^{(t)} + b_2) \in R^{|V|}$$

- Скрытые состояния

$$h^{(t)} = \sigma(W_h h^{(t-1)} + W_x e^{(t-1)} + b_1)$$

- Эмбеддинги слов

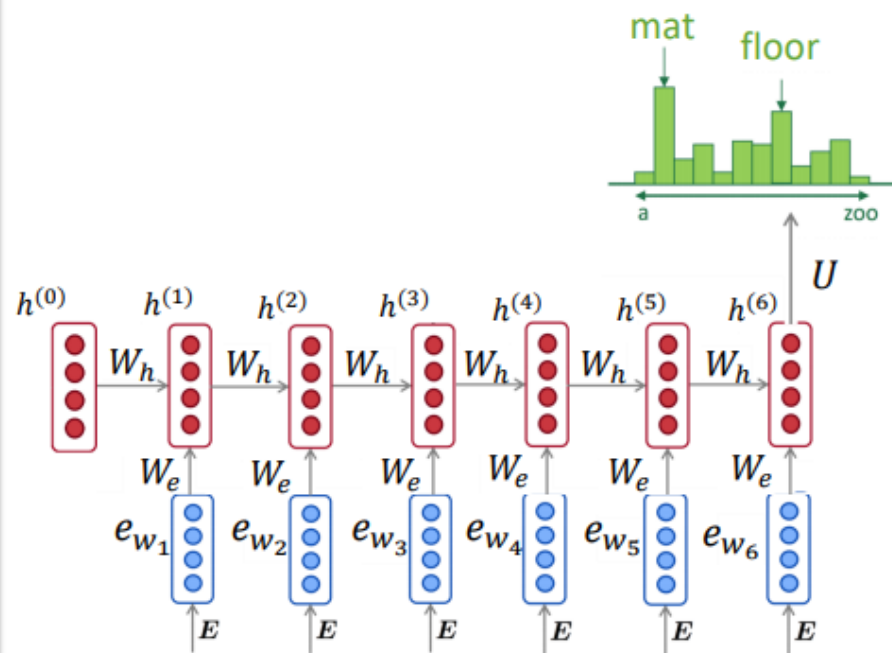
- Слова или токены



$\langle bos \rangle$ Котик очень тихо спал на
 $x^{(i-6)} \quad x^{(i-5)} x^{(i-4)} \quad x^{(i-3)} \quad x^{(i-2)} \quad x^{(i-1)}$

Neural Language Models (NLM)

- Рекуррентные языковые модели (RNN LM)
- Какие **улучшения** в сравнении с предыдущими моделями?



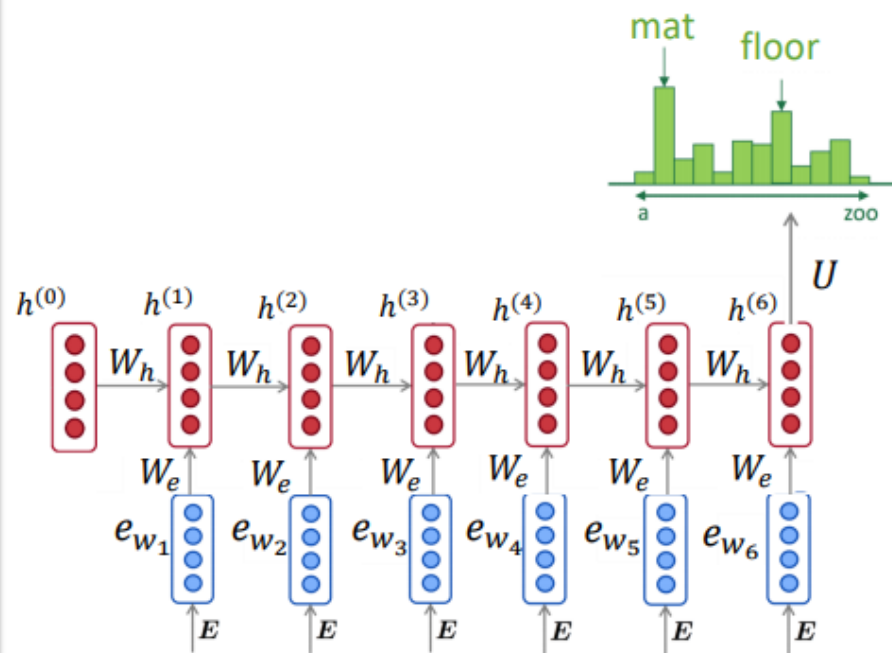
$\langle bos \rangle$ Котик очень тихо спал на
 $x^{(i-6)} \quad x^{(i-5)} x^{(i-4)} \quad x^{(i-3)} \quad x^{(i-2)} \quad x^{(i-1)}$

Neural Language Models (NLM)

- Рекуррентные языковые модели (RNN LM)

- Какие **улучшения** в сравнении с предыдущими моделями:

- Можем обрабатывать последовательности любой длины
- Размер модели не зависит от размера входа
- Вычисления для шага t (в теории) зависят от множества предыдущих шагов
- Представления зависят от шагов



$\langle bos \rangle$ Котик очень тихо спал на
 $x^{(i-6)} \quad x^{(i-5)} x^{(i-4)} \quad x^{(i-3)} \quad x^{(i-2)} \quad x^{(i-1)}$

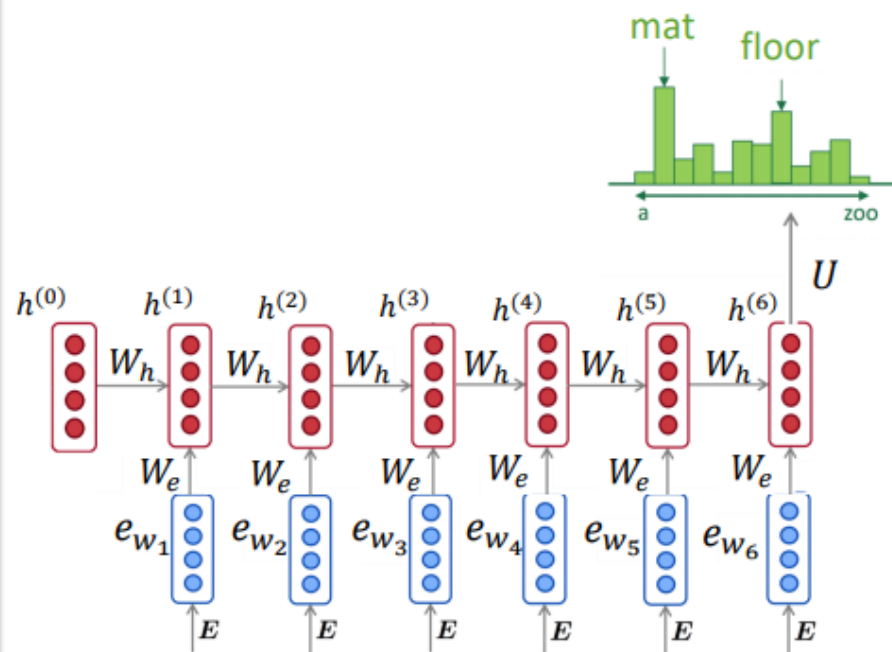
Neural Language Models (NLM)

- Рекуррентные языковые модели (RNN LM)

- Какие **улучшения** в сравнении с предыдущими моделями:

- Можем обрабатывать последовательности любой длины
- Размер модели не зависит от размера входа
- Вычисления для шага t (в теории) зависят от множества предыдущих шагов
- Представления зависят от шагов

- Какие **проблемы**?



$\langle bos \rangle$ Котик очень тихо спал на
 $x^{(i-6)} \quad x^{(i-5)} x^{(i-4)} \quad x^{(i-3)} \quad x^{(i-2)} \quad x^{(i-1)}$

Neural Language Models (NLM)

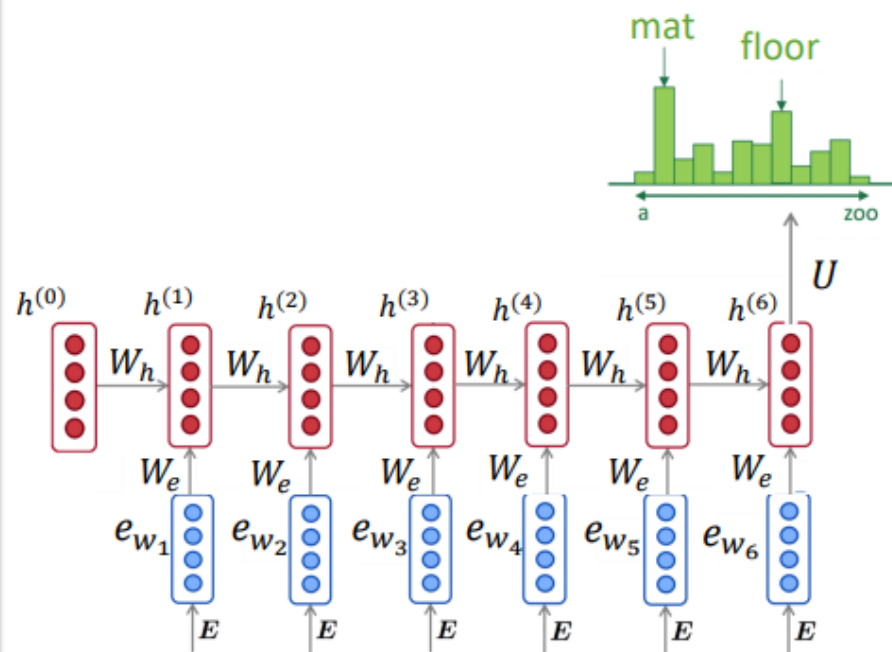
- Рекуррентные языковые модели (RNN LM)

- Какие **улучшения** в сравнении с предыдущими моделями:

- Можем обрабатывать последовательности любой длины
- Размер модели не зависит от размера входа
- Вычисления для шага t (в теории) зависят от множества предыдущих шагов
- Представления зависят от шагов

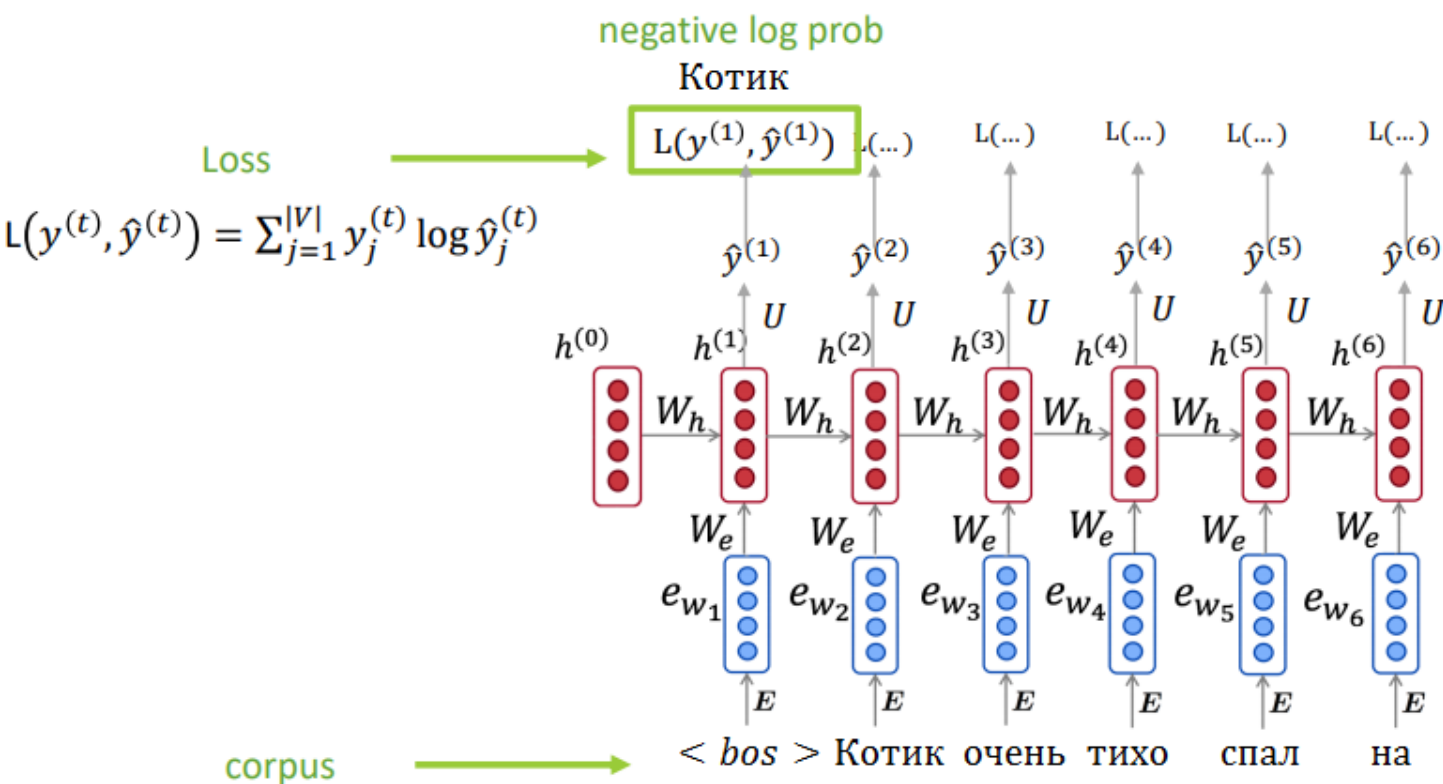
- Какие **проблемы**:

- Вычисления могут быть медленны
- На практике сеть может не иметь доступа к информации на много шагов назад

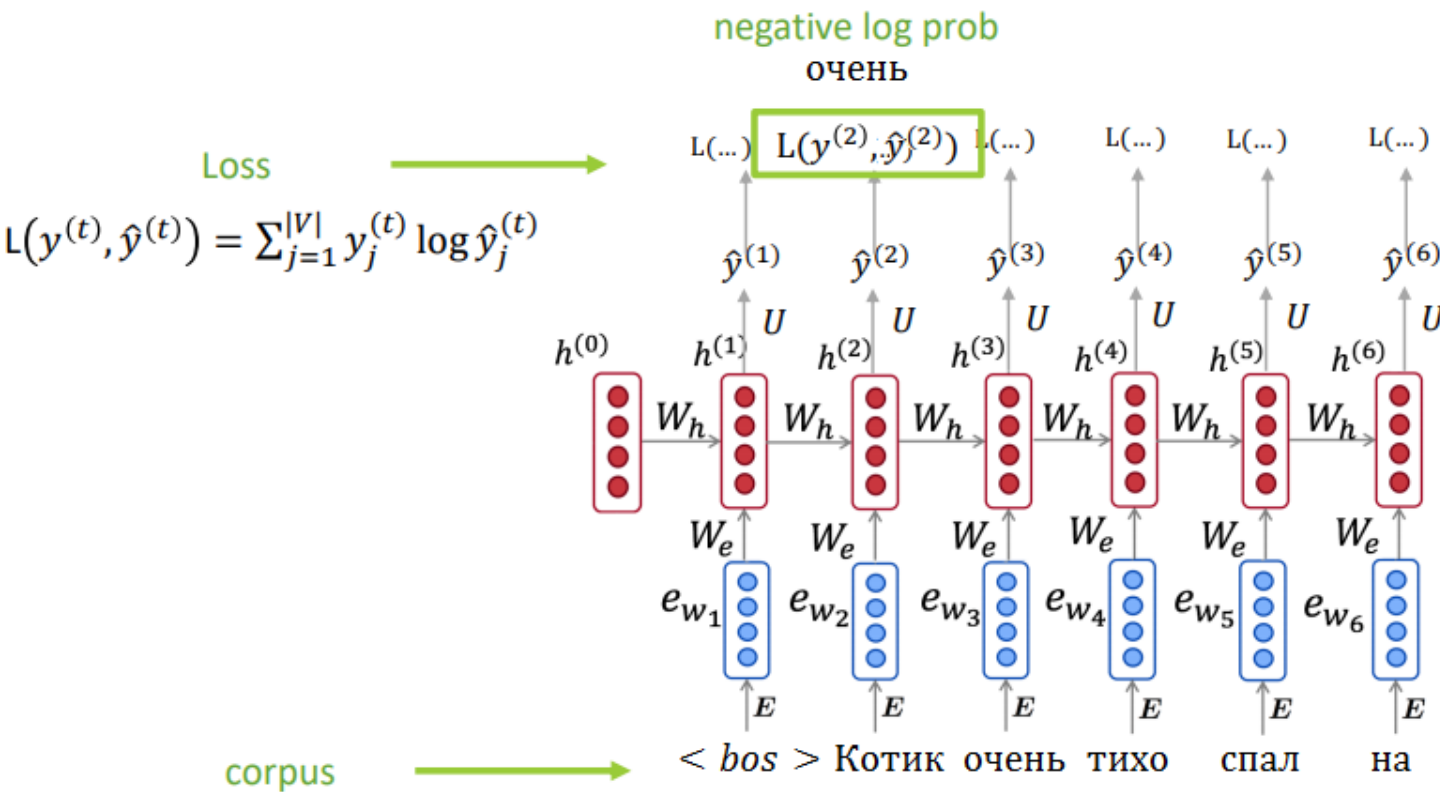


$\langle bos \rangle$ Котик очень тихо спал на
 $x^{(i-6)} \quad x^{(i-5)} x^{(i-4)} \quad x^{(i-3)} \quad x^{(i-2)} \quad x^{(i-1)}$

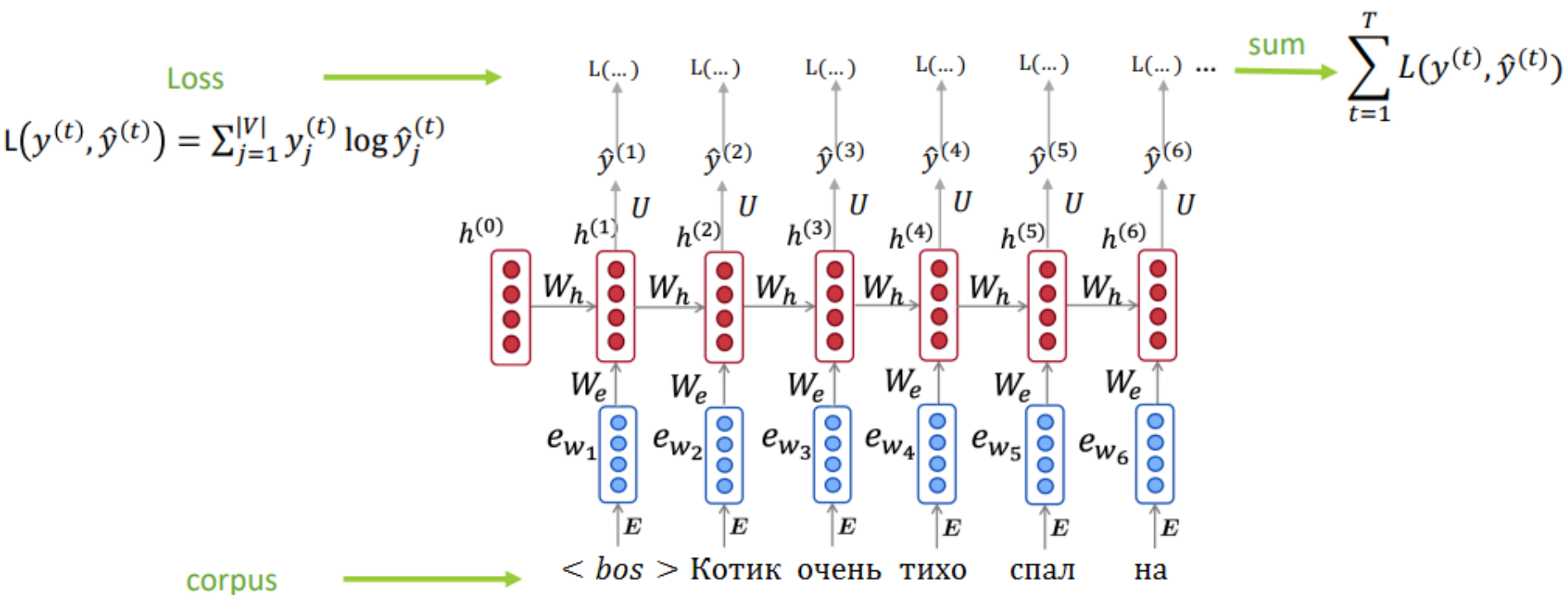
Training RNN-LM



Training RNN-LM



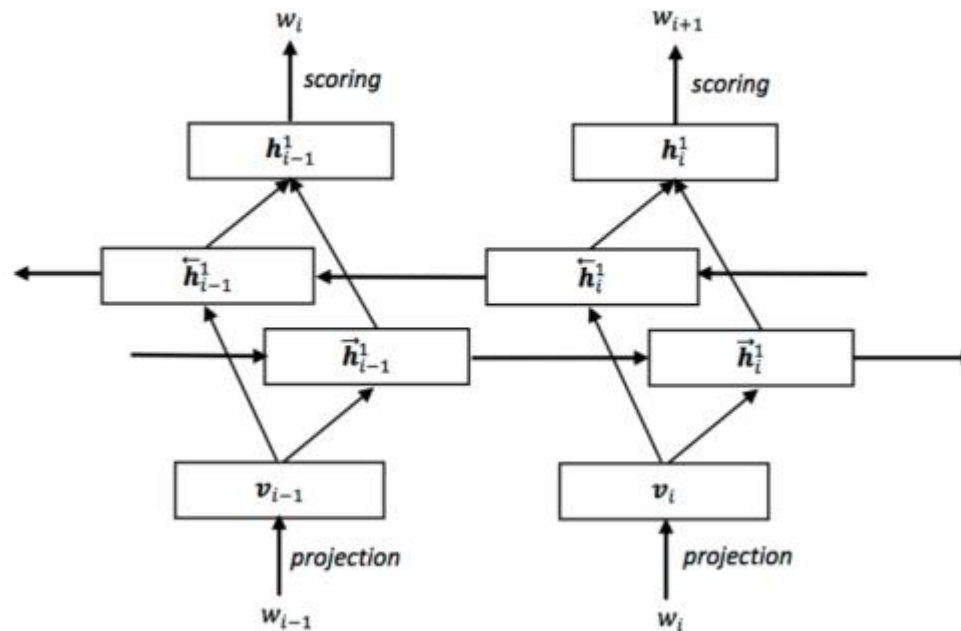
Training RNN-LM



- Языковые модели можно разделить на два типа:
 - **Count-based Models (Statistical)** – статистические языковые модели.
 - Марковское предположение порядка n ;
 - Аппроксимация вероятностей n -gram (counting and smoothing).
 - **Neural Language Models** – языковые модели на основе нейронных сетей.
 - Решена проблема разреженности N -gram модели представлением слов с помощью векторов? – Частично.
 - Параметры слов являются частью процесса обучения модели.

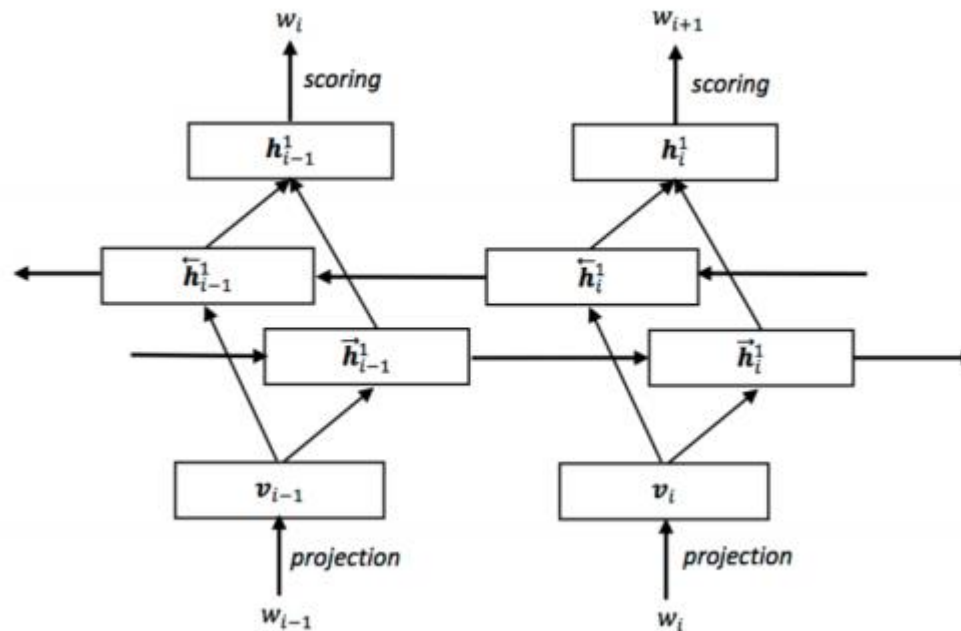
Двунаправленные языковые модели

- Можем использовать как левый, так и правый контексты.
- **Не можем** использовать для генерации ☹️.
- **Когда же использовать** такую языковую модель?



Двунаправленные языковые модели

- Можем использовать как левый, так и правый контексты.
- **Не можем** использовать для генерации ☹️.
- **Когда же использовать** такую языковую модель – в других задачах 😊, например исправление опечаток, NER и т. д.



Сверточные языковые модели

- Также фиксированное окно!
- Но вместо конкатенации используются сверточные слои.

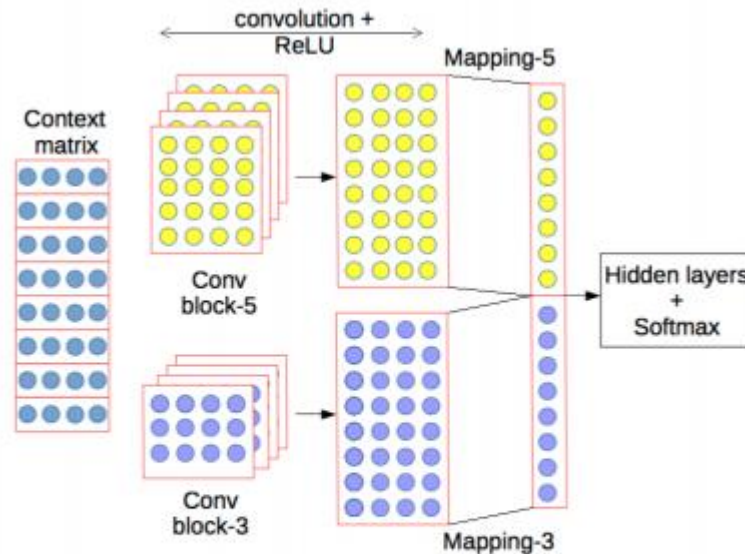


Figure 3: Combining kernels with different sizes. We concatenate the outputs of 2 convolutional blocks with kernel size of 5 and 3 respectively.

Языковые модели как эмбединги

- Deep contextualized word representations (ELMO).
- Полный эмбединг слова: конкатенация word embedding и char-cnn.
- Обучаем Bi-LSTM на большом корпусе.
- Используем взвешенную сумму скрытых представлений с разных слоев.
- В чем улучшение?



(ELMO – Embeddings from Language MOdel)

Языковые модели как эмбединги

- Deep contextualized word representations (ELMO).
- Полный эмбединг слова: конкатенация word embedding и char-cnn.
- Обучаем Bi-LSTM на большом корпусе.
- Используем взвешенную сумму скрытых представлений с разных слоев.
- **В чем улучшение?**
 - Используем символьную информацию.
 - Используем «весь» контекст последовательности.



(ELMO – Embeddings from Language MOdel)

Языковые модели как эмбединги

- Universal Language Model Fine-tuning (ULMFiT).

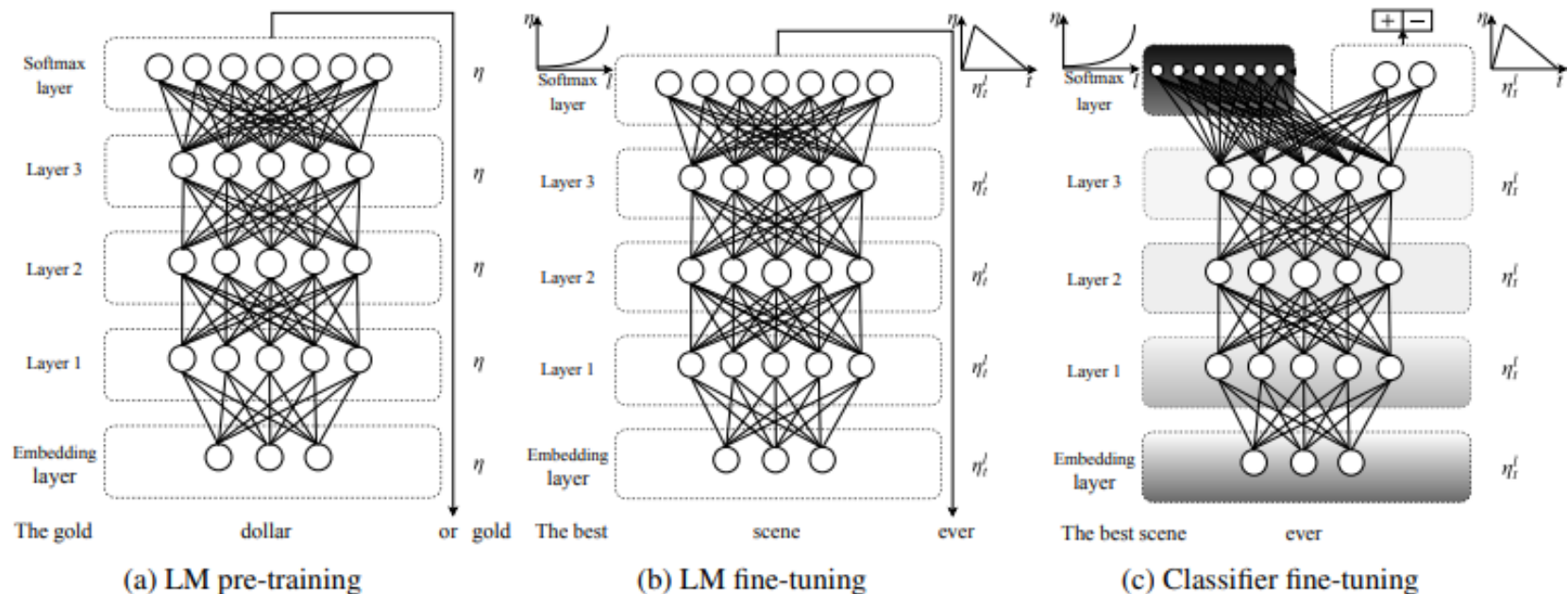


Figure 1: ULMFiT consists of three stages: a) The LM is trained on a general-domain corpus to capture general features of the language in different layers. b) The full LM is fine-tuned on target task data using discriminative fine-tuning ('Discr') and slanted triangular learning rates (STLR) to learn task-specific features. c) The classifier is fine-tuned on the target task using gradual unfreezing, 'Discr', and STLR to preserve low-level representations and adapt high-level ones (shaded: unfreezing stages; black: frozen).

Языковые модели как эмбединги

- Contextual String Embeddings for Sequence Labeling.

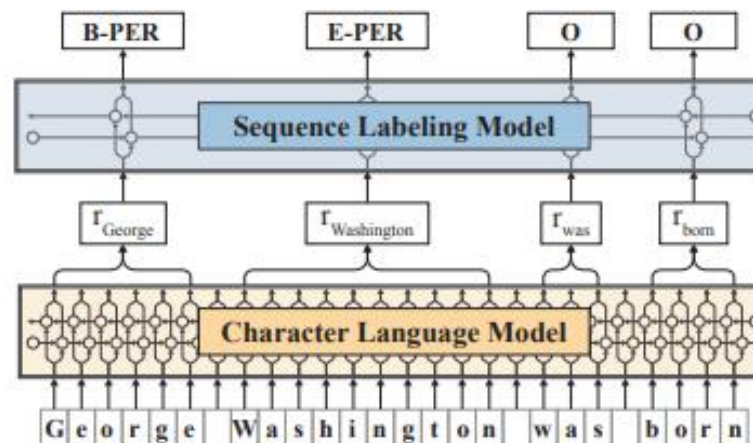


Figure 1: High level overview of proposed approach. A sentence is input as a character sequence into a pre-trained bidirectional character language model. From this LM, we retrieve for each word a contextual embedding that we pass into a vanilla BiLSTM-CRF sequence labeler, achieving robust state-of-the-art results on downstream tasks (NER in Figure).

- [AWD-LSTM](#) (2017)
- [ELMO](#) (март 2018)
- [ULMFit](#) (май 2018)
- [Contextual String Embeddings](#) (август 2018)
- [BERT](#) (октябрь 2018)

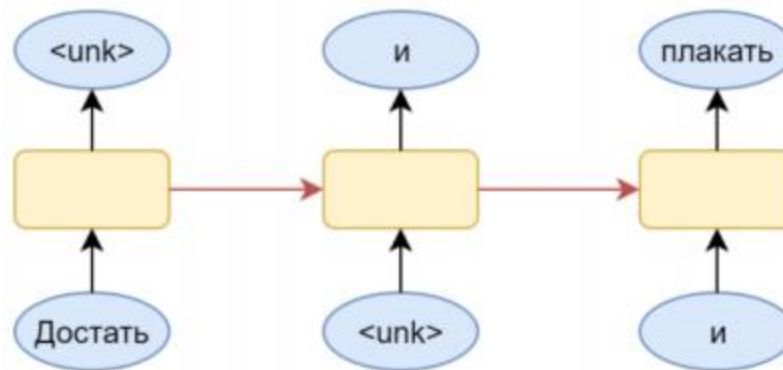
Генерация стихов

Обработка отсутствующих слов в RNN LM

- Словарь может состоять из миллионов слов, но часто его приходится сильно фильтровать.
- **Проблема**: как обрабатывать неизвестные слова?

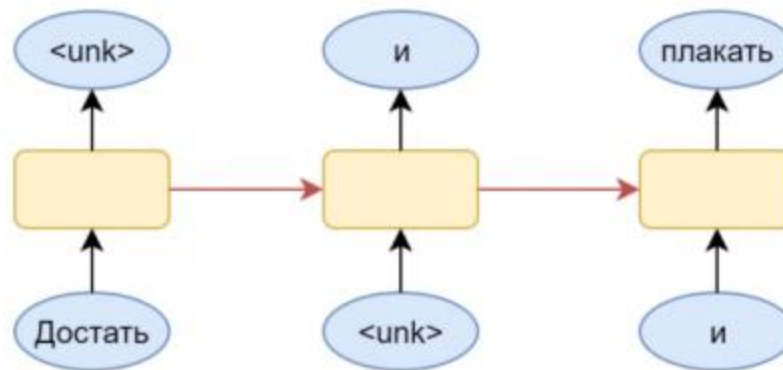
Обработка отсутствующих слов в RNN LM

- Словарь может состоять из миллионов слов, но часто его приходится сильно фильтровать.
- Проблема:** как обрабатывать неизвестные слова?
- Вместо отсутствующего слова берём <unk>:



Обработка отсутствующих слов в RNN LM

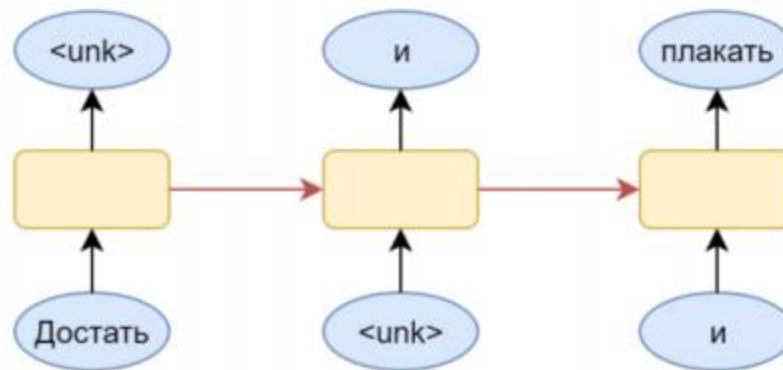
- Словарь может состоять из миллионов слов, но часто его приходится сильно фильтровать.
- Проблема:** как обрабатывать неизвестные слова?
- Вместо отсутствующего слова берём <unk>:



- В модели предсказания слова по предыдущему выдаваемое распределение на словах сместится в пользу <unk>.

Обработка отсутствующих слов в RNN LM

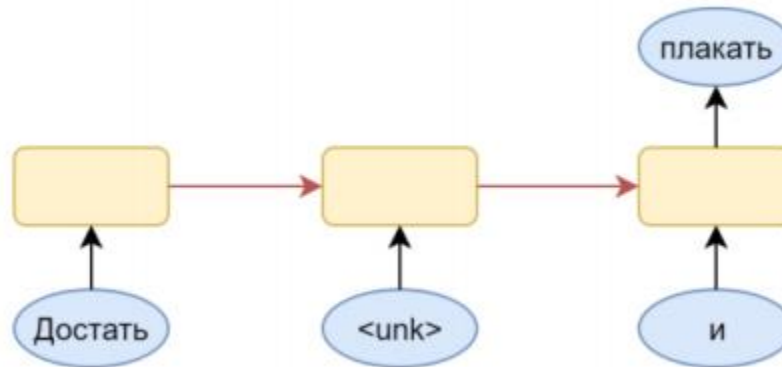
- Словарь может состоять из миллионов слов, но часто его приходится сильно фильтровать.
- Проблема:** как обрабатывать неизвестные слова?
- Вместо отсутствующего слова берём <unk>:



- В модели предсказания слова по предыдущему выдаваемое распределение на словах сместится в пользу <unk>.
- Решение:** можно семплировать без слова <unk>, но получается «криво».

Обработка отсутствующих слов в RNN LM

- Словарь может состоять из миллионов слов, но часто его приходится сильно фильтровать.
- Проблема:** как обрабатывать неизвестные слова?
- Альтернатива – предсказывать слова по цепочке предыдущих:



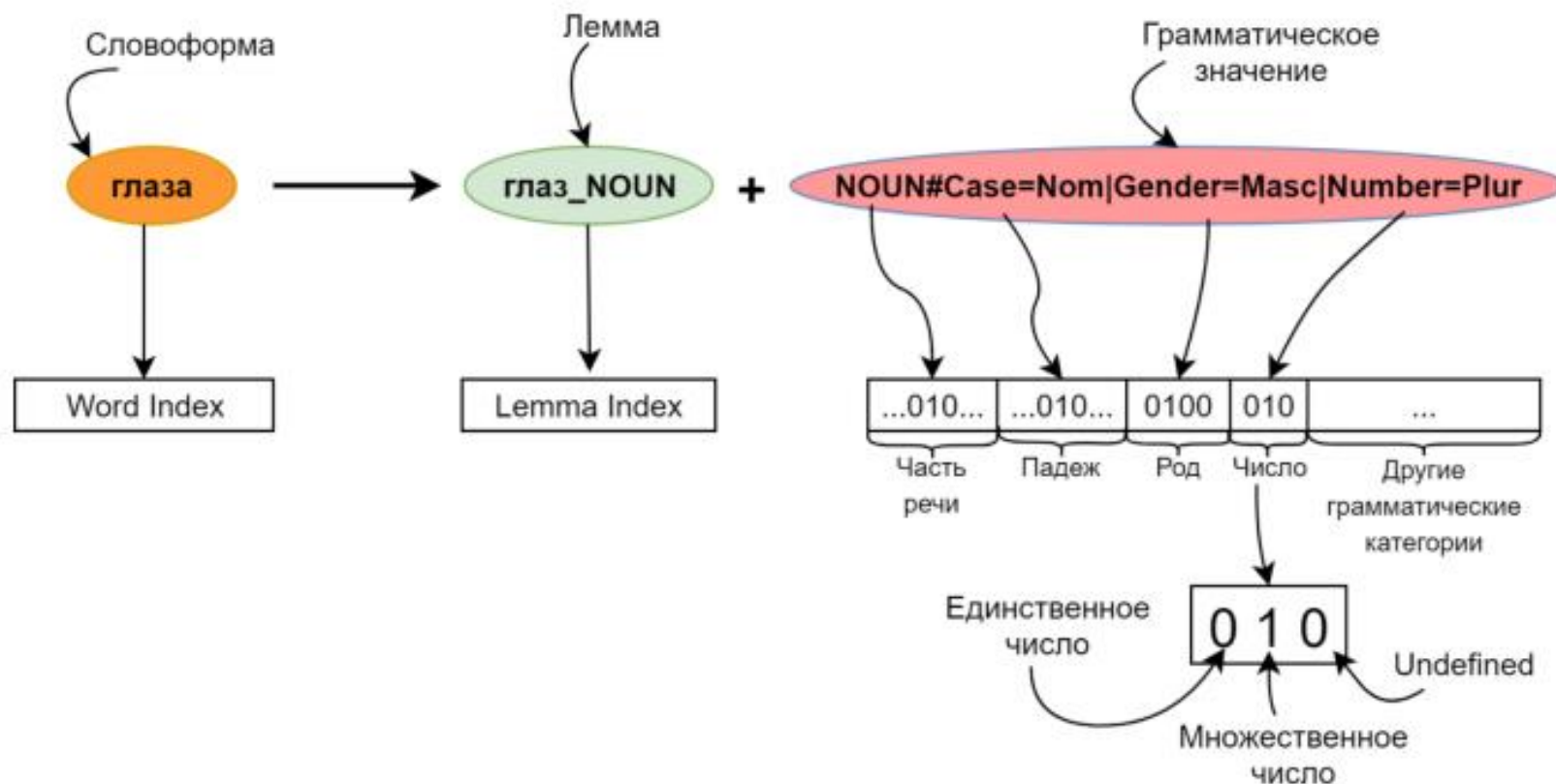
- Из обучающей выборки придётся нарезать всевозможные цепочки, что приведёт к её существенному увеличению
- Зато можно выкинуть все цепочки, заканчивающиеся неизвестным словом.

Доработка входного/выходного слоя

- Необходимо сократить размерность выходного слоя.

Доработка входного/выходного слоя

- Необходимо сократить размерность выходного слоя.



- Можно использовать уже предобученные эмбединги для лемм. Например от [RusVectors](https://rusvectors.github.io/).

Доработка входного/выходного слоя

- Вместо индекса слова можно предсказывать по-отдельности лемму и грамматическое значение.

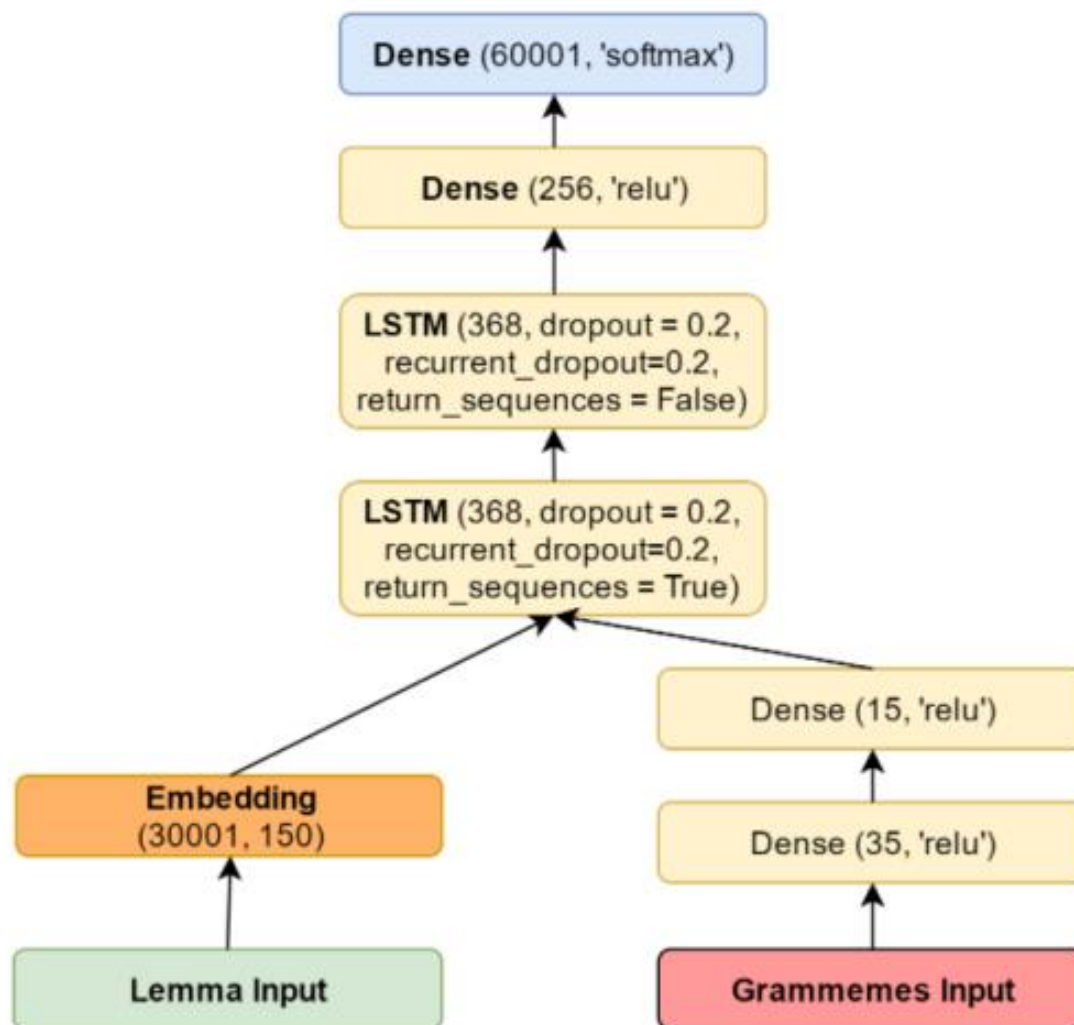
Доработка входного/выходного слоя

- Вместо индекса слова можно предсказывать по-отдельности лемму и грамматическое значение.
- **Проблема:** у сэмплированной леммы может не оказаться нужного грамматического значения.

Доработка входного/выходного слоя

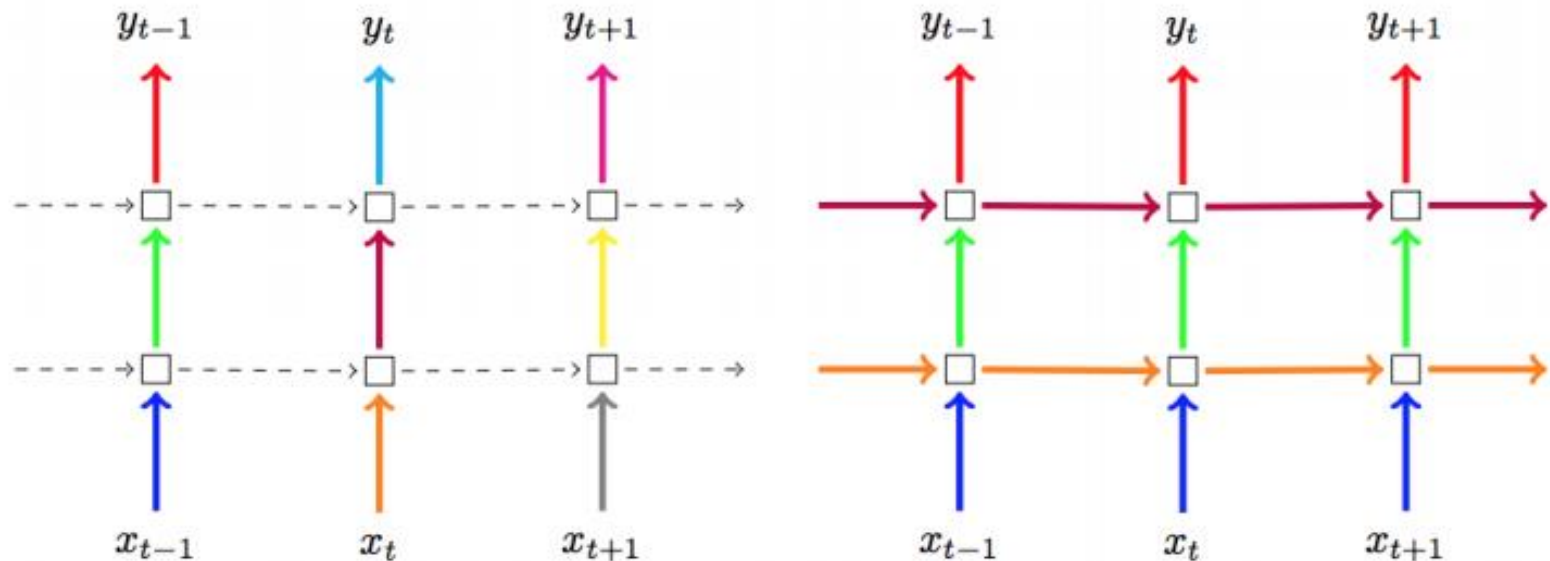
- Вместо индекса слова можно предсказывать по-отдельности лемму и грамматическое значение.
- **Проблема:** у сэмплированной леммы может не оказаться нужного грамматического значения.
- **Варианты решения:**
 - Выбирать наиболее вероятную пару «лемма + грамматическое значение» из существующих.
 - Выбирать наиболее вероятное грамматическое значение среди возможных для сэмплированной леммы.

Итоговая архитектура сети (Keras)

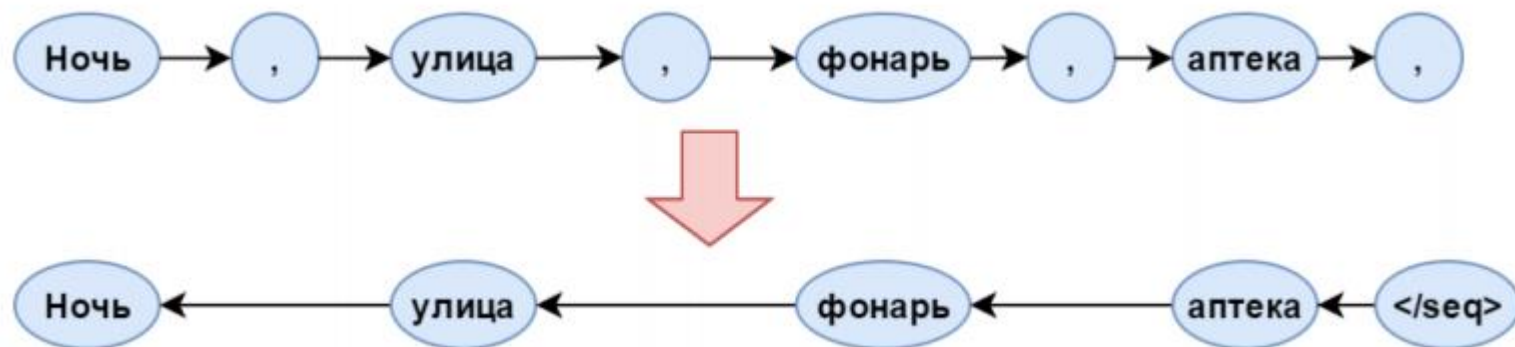


Рекуррентный dropout

- Обычный dropout применяется к весам, связанным с входными данными.
- Можно применять dropout и к весам, связанным с выходами предыдущего шага h_{t-1} .



- <http://stihi.ru/> + морфологическая разметка.
- Объект выборки – строка стихотворения.
- В конец каждой строки добавлялся завершающий символ.
- Строки инвертировались для упрощения рифмовки при генерации.
- Из выборки удалены знаки препинания (сеть сильно обучается на запятых и многоточиях).

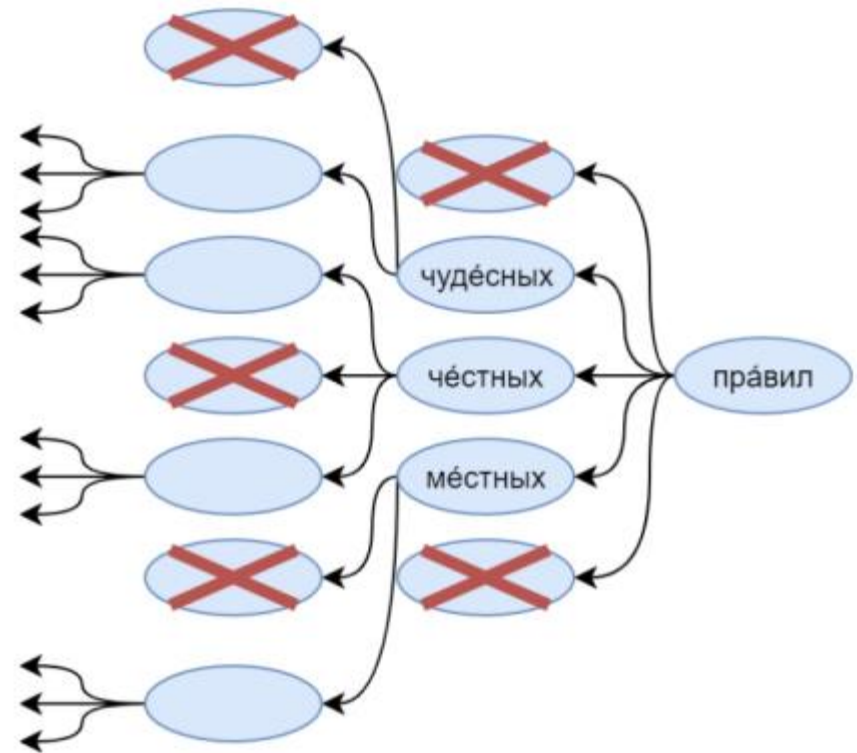


Правила фильтрации

- У нас есть модель-генератор, нужно фильтровать слова так, чтобы получались именно стихотворения.
- Метрические правила определяют последовательность ударных и безударных слогов в строке.
- Правила рифмы допускают только словоформы, которые корректно рифмуются (слова с одной леммой рифмовать запрещено).
- Ударения были получены путём обучения классификатора на словаре, рифмы – эвристическими правилами.

Лучевой поиск (beam search)

- В результате работы фильтров могло не остаться ни одного слова.
- Для борьбы с этим на каждом этапе применения фильтров берём не лучшего, а несколько лучших кандидатов, так, чтобы на каждом шаге их было N штук.



*Так толку мне теперь грустить
Что будет это прожито
Не суждено кружить в пути
Почувствовав боль бомжика*



СПАСИБО ЗА ВНИМАНИЕ

- NLP курс в яндексе https://github.com/yandexdataschool/nlp_course
- Курс в Stanford CS224N <http://cs224n.stanford.edu/>
- Презентации Екатерины Черняк <https://github.com/echernyak/nlp-course-fintech/>