

Языковые модели и Transfer Learning

О чем мы сегодня поговорим

- Языковые модели
 - Что такое языковая модель
 - N-граммные модели
 - Нейронные модели
 - Стратегии генерации текста
- Transfer Learning
 - ELMo
 - BERT

Примеры из жизни

- Подсказки в поисковике
- Модель исправляющая ошибки

где курить елку



Я люблю кушать

- я люблю кушать
- я люблю кушать **на английском**
- я люблю кушать **песня**
- я люблю кушать **мертвых людей**
- я люблю кушать **на корейском**
- я люблю кушать **яблоко перевод на английский**
- я люблю кушать **на чеченском**
- я люблю **покушать** цитаты
- я люблю кушать **перевод**
- я люблю кушать **что делать**

Пожаловаться на неприемлемые подсказки

Все

Картинки

Видео

Покупки

Новости

Ещё

Настройки

Инструменты

Результатов: примерно 43 600 000 (1,01 сек.)

Показаны результаты по запросу **где *купить* елку**

Искать вместо этого **где курить елку**

Что такое языковая модель?

- Языковые модели (LM) оценивают вероятность различных лингвистических единиц: символов, токенов, последовательностей токенов
- Цель - оценить вероятности появления фрагментов текста

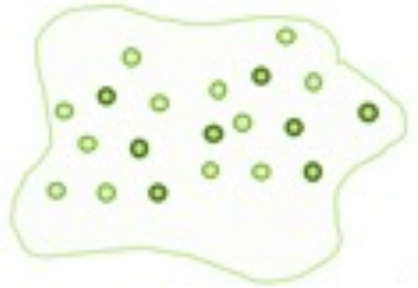
Что такое языковая модель?

Предположим, что мы имеем дело с предложениями
Хочется, чтобы вероятности отражали знание языка



Предложения, которые «с большей вероятностью» появятся в языке, имели большую вероятность в соответствии с нашей языковой моделью

Что если оценить вероятности?



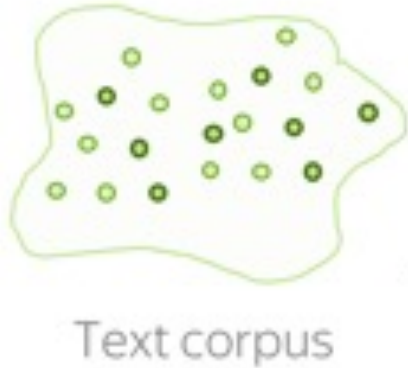
Text corpus

$$P(\text{Я не просрочил дедлайны}) = 0 / |\text{корпус}| = 0$$

$$P(\text{Просрочил дедлайны я не}) = 0 / |\text{корпус}| = 0$$

- Предложения, которые мы никогда не встречали в обучающем корпусе, будут иметь нулевую вероятность
- Первое предложение более вероятно, чем второе, но вероятности у них одинаковые

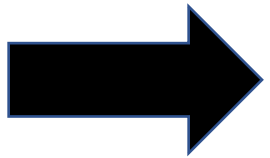
Что если оценить вероятности?



$$P(\text{Я не просрочил дедлайны}) = 0 / |\text{корпус}| = 0$$

$$P(\text{Просрочил дедлайны я не}) = 0 / |\text{корпус}| = 0$$

- Предложения, которые мы никогда не встречали в обучающем корпусе, будут иметь нулевую вероятность
- Первое предложение более вероятно, чем второе, но вероятности у них одинаковые



Декомпозируем предложения и разложим вероятность предложения на вероятности более мелких частей

Используем chain rule

Токены

$$P(y_1, y_2, \dots, y_n) = P(y_1) \cdot P(y_2|y_1) \cdot P(y_3|y_1, y_2) \cdot \dots \cdot P(y_n|y_1, \dots, y_{n-1}) = \prod_{t=1}^n P(y_t|y_{<t}).$$

Вероятность увидеть
токены в таком порядке

$P(\mathbf{I}) =$

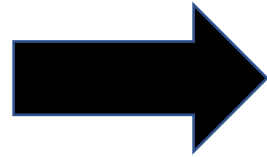
$P(\mathbf{I})$

└──

Probability of \mathbf{I}

Как оценить $P(y_t | y_1, y_2, \dots, y_{t-1})$?

- N-граммные модели
- Нейронные модели



Отличаются только
способом считать
вероятности

N-gram модели

- Посчитаем вероятности по текстовому корпусу

$$P(y_t | y_1, y_2, \dots, y_{t-1}) = \frac{N(y_1, y_2, \dots, y_t)}{N(y_1, y_2, \dots, y_{t-1})},$$

Кол-во раз, которое
последовательность

(y_1, y_2, \dots, y_t)

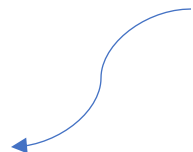
встречается в тексте

N-gram модели

- Посчитаем вероятности по текстовому корпусу

$$P(y_t | y_1, y_2, \dots, y_{t-1}) = \frac{N(y_1, y_2, \dots, y_t)}{N(y_1, y_2, \dots, y_{t-1})},$$

Кол-во раз, которое
последовательность
 (y_1, y_2, \dots, y_t)
встречается в тексте



- Та же проблема что и с целыми предложениями – длинные последовательности маловероятно встретятся в тексте

N-gram модели

- Марковское предположение (Markov assumption)

Вероятность следующего слова зависит только от фиксированного количества предыдущих слов

$$P(y_t | \underbrace{y_1, y_2, \dots, y_{t-1}}_{\text{all previous tokens}}) = P(y_t | \underbrace{y_{t-n+1}, \dots, y_{t-1}}_{\text{n-1 previous token}})$$

all previous tokens

n-1 previous token

n=3 (trigram model): $P(y_t | y_1, y_2, \dots, y_{t-1}) = P(y_t | y_{t-2}, y_{t-1})$

n=2 (bigram model): $P(y_t | y_1, y_2, \dots, y_{t-1}) = P(y_t | y_{t-1})$

n=1 (unigram model): $P(y_t | y_1, y_2, \dots, y_{t-1}) = P(y_t)$

Частота слова ☺

N-gram модели

Before

$P(\text{I saw a cat on a mat}) =$

$P(\text{I})$

- $P(\text{saw} | \text{I})$
- $P(\text{a} | \text{I saw})$
- $P(\text{cat} | \text{I saw a})$
- $P(\text{on} | \text{I saw a cat})$
- $P(\text{a} | \text{I saw a cat on})$
- $P(\text{mat} | \text{I saw a cat on a})$

After (3-gram)

$P(\text{I saw a cat on a mat}) =$

$P(\text{I})$

- $P(\text{saw} | \text{I})$
- $P(\text{a} | \text{I saw})$
- $P(\text{cat} | \text{I saw a})$
- $P(\text{on} | \text{I saw a cat})$
- $P(\text{a} | \text{I saw a cat on})$
- $P(\text{mat} | \text{I saw a cat on a})$

→ $P(\text{I})$

→ • $P(\text{saw} | \text{I})$

→ • $P(\text{a} | \text{I saw})$

→ • $P(\text{cat} | \text{saw a})$

→ • $P(\text{on} | \text{a cat})$

→ • $P(\text{a} | \text{cat on})$

→ • $P(\text{mat} | \text{on a})$

ignore use

Васкофф сглаживание

$$P(\text{mat} \mid \text{I saw a cat on a}) = P(\text{mat} \mid \text{cat on a}) = \frac{N(\text{cat on a mat})}{N(\text{cat on a})}$$

$$P(\text{mat} \mid \text{cat on a}) = \frac{N(\text{cat on a mat})}{\boxed{N(\text{cat on a})}} = ?$$

zero

not good: can not compute the probability

$$P(\text{mat} \mid \text{cat on a}) = \frac{\overset{\text{zero}}{\boxed{N(\text{cat on a mat})}}}{N(\text{cat on a})} = ?$$

not good: zeros out probability of the sentence

Backoff сглаживание

Давайте использовать меньший контекст, для контекстов, о которых мы мало знаем:

- Если можем, то триграммы
- Если нет, то биграммы
- Если нет, то униграммы

$N(\text{cat on a}) = 0 \longrightarrow \text{try "on a"}$

$P(\text{mat} \mid \text{cat on a}) \approx P(\text{mat} \mid \text{on a})$

$N(\text{on a}) = 0 \longrightarrow \text{try "a"}$

$P(\text{mat} \mid \text{on a}) \approx P(\text{mat} \mid \text{a})$

$N(\text{a}) = 0 \longrightarrow \text{try unigram}$

$P(\text{mat} \mid \text{a}) \approx P(\text{mat})$

Линейная интерполяция

$$\begin{aligned}\widehat{P}(\text{mat} \mid \text{cat on } a) \approx & \lambda_3 P(\text{mat} \mid \text{cat on } a) + \\ & \lambda_2 P(\text{mat} \mid \text{on } a) + \\ & \lambda_1 P(\text{mat} \mid a) + \\ & \lambda_0 P(\text{mat})\end{aligned}$$

$$\sum_i \lambda_i = 1$$

Сглаживание Лапласа

Pretend we saw each n-gram at least one time (or δ times):

$$\hat{P}(\text{mat} \mid \text{cat on a}) = \frac{\delta + N(\text{cat on a mat})}{\delta \cdot |V| + N(\text{cat on a})}$$

Нейронная модель

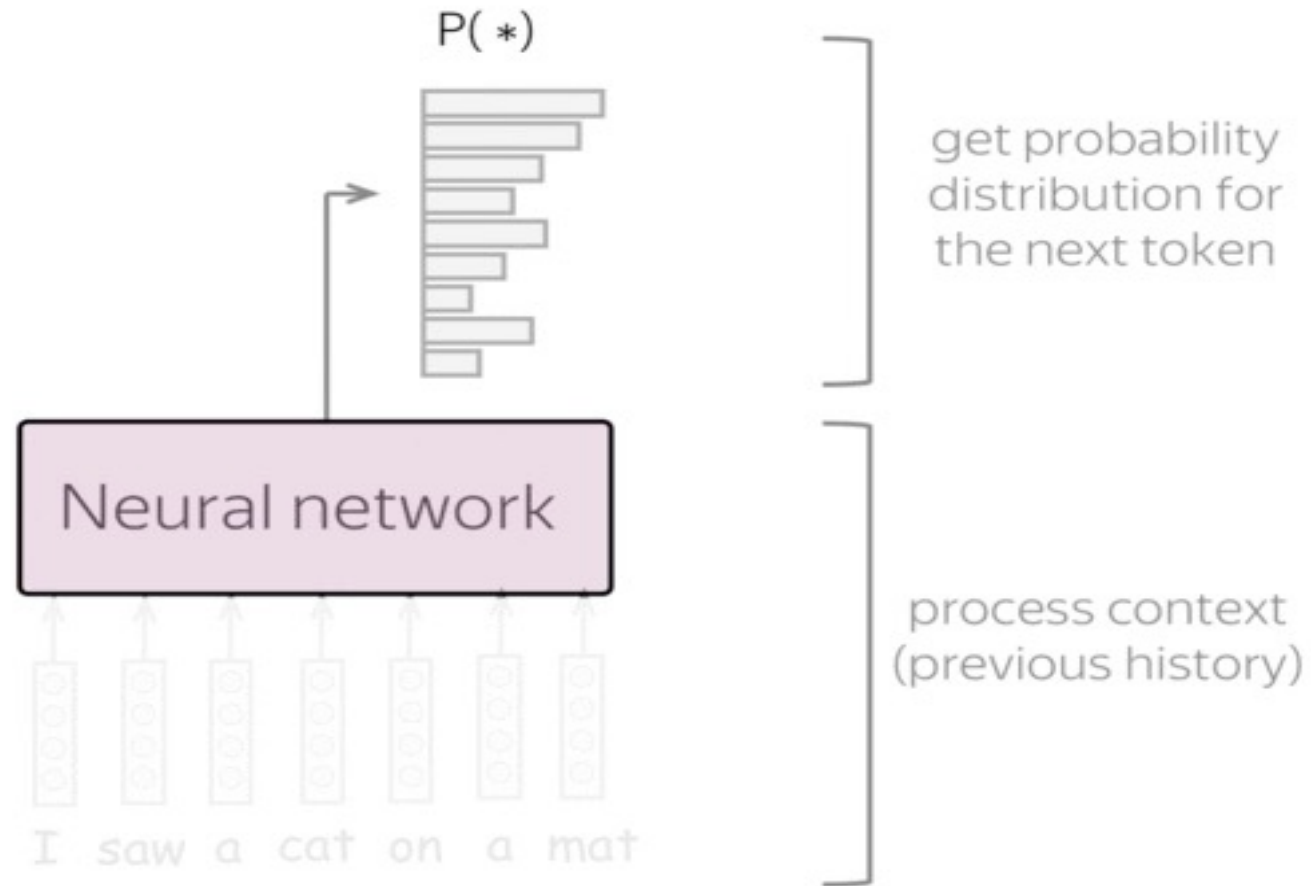
Обучим нейронную сеть, чтобы предсказывать вероятности:

- Нужен representation вектор для контекста, который мы уже видели
- Нужно сгенерировать распределение вероятностей по всем токенам в тексте

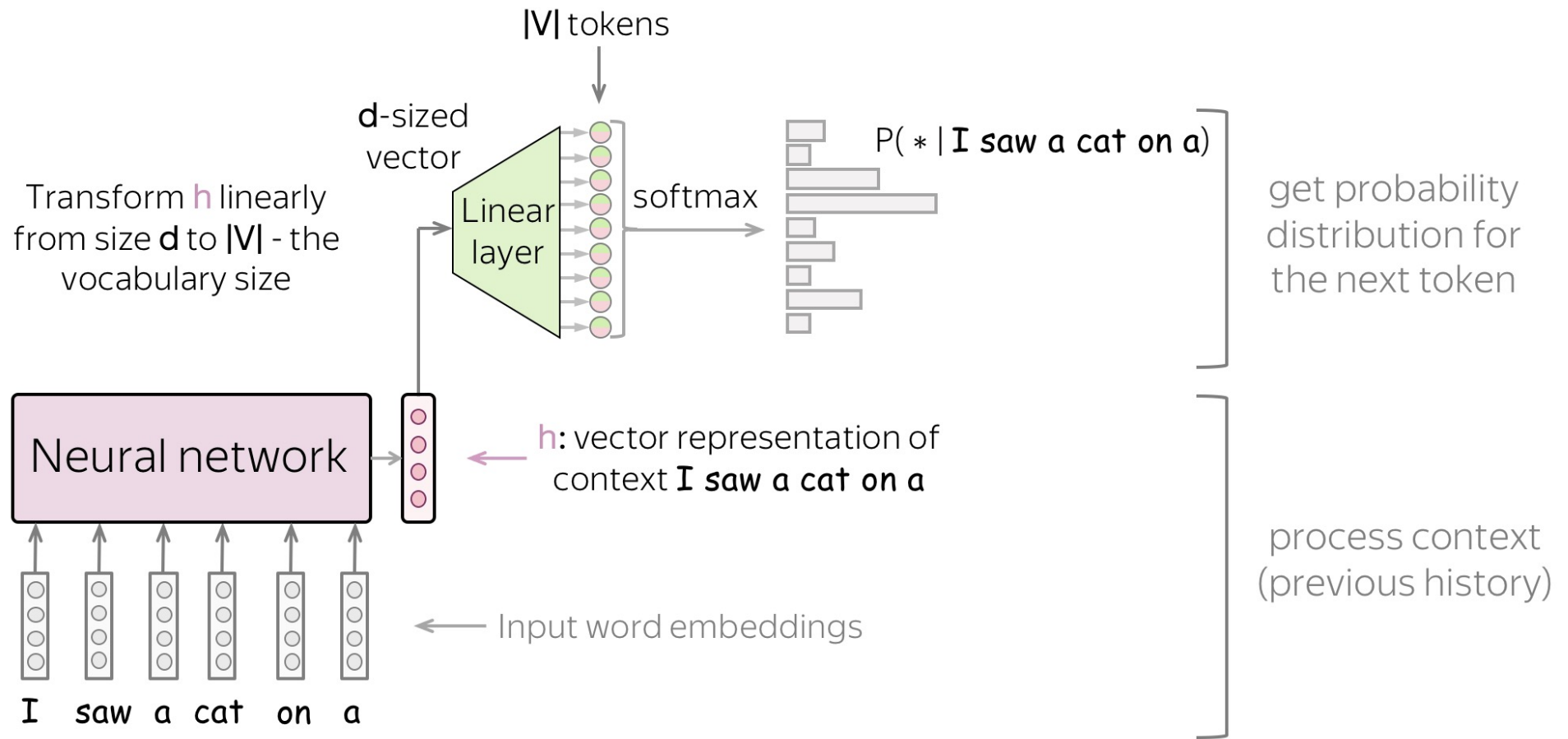


Похоже на то что мы делали в
классификации текстов

Нейронная модель



Нейронная модель



Как генерировать текст?

- Greedy decoding

На каждом шаге выбираем токен с наибольшей вероятностью

Как генерировать текст?

- Greedy decoding

На каждом шаге выбираем токен с наибольшей вероятностью

Минусы:

- быстро зациклимся на каком-то слове
- для конкретного начала будет только одно продолжение
- будем использовать небольшое количество слов из корпуса

Как генерировать текст?

- Greedy decoding

На каждом шаге выбираем токен с наибольшей вероятностью

Минусы:

- быстро зациклимся на каком-то слове
- для конкретного начала будет только одно продолжение
- будем использовать небольшое количество слов из корпуса



Хочется видеть **связный** и
разнообразный текст

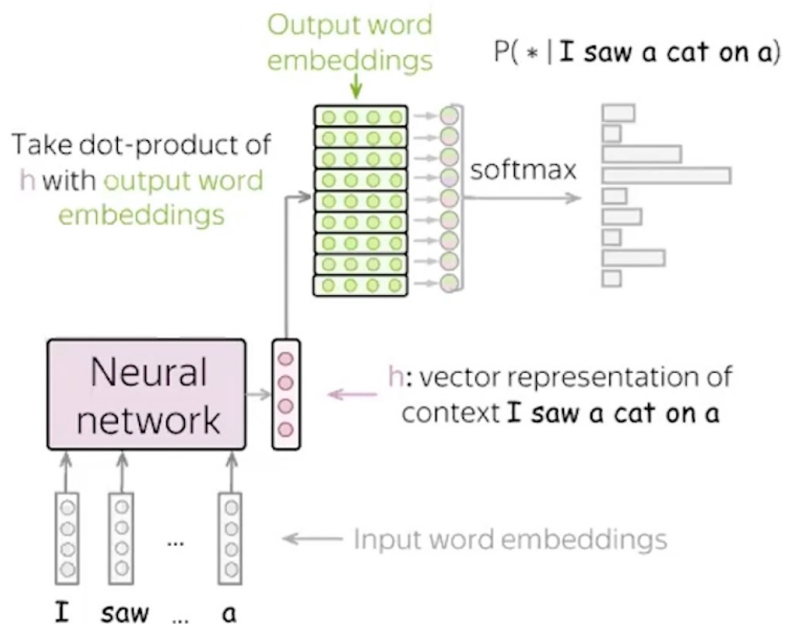
Как генерировать текст?

- Сэмплирование с температурой

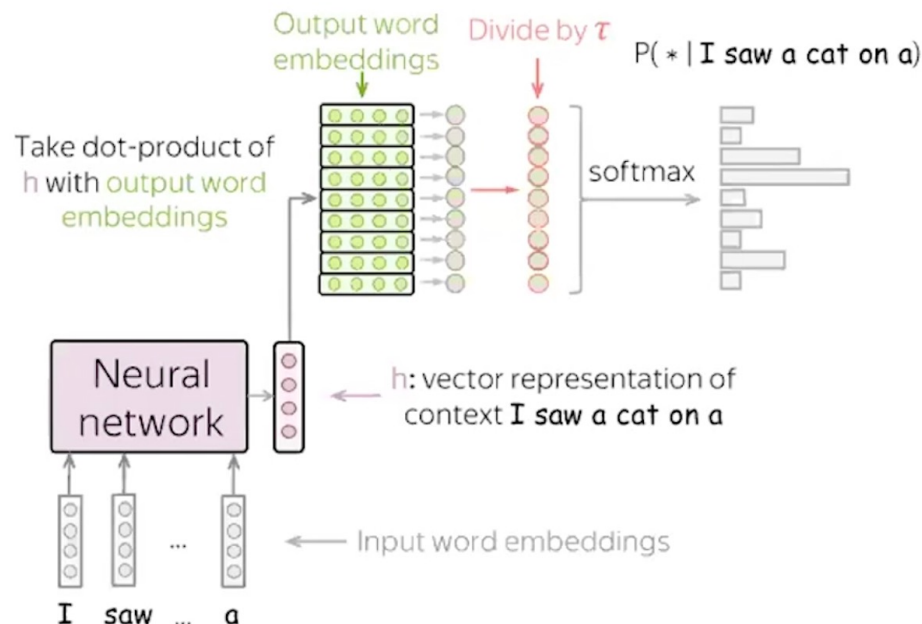
$$\frac{\exp(h^T w)}{\sum_{w_i \in V} \exp(h^T w_i)} \rightarrow \frac{\exp\left(\frac{h^T w}{\tau}\right)}{\sum_{w_i \in V} \exp\left(\frac{h^T w_i}{\tau}\right)}$$

τ - softmax temperature

Before



After



Как генерировать текст?

- Top-K sampling

Будем сэмплировать из k фиксированных наиболее вероятных токенов

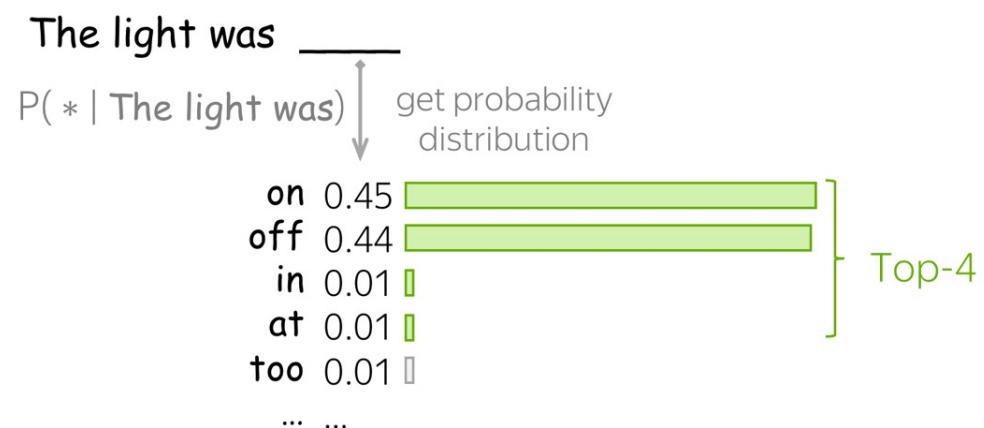
Как генерировать текст?

- Top-K sampling

Top-K for a flat distribution: not enough



Top-K for a peaky distribution: too many



Как генерировать текст?

- Top-K sampling

Будем сэмплировать из k фиксированных наиболее вероятных токенов

Минусы:

- покроем только небольшую часть общей вероятностной массы
- может содержать маловероятные токены

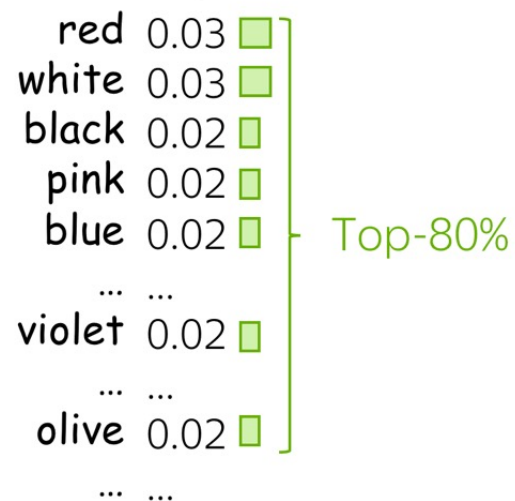
Как генерировать текст?

- Nucleus sampling

Будем сэмплировать из top-p% вероятностной массы

The dress color was _____

$P(* | \text{The dress color was})$



The light was _____

$P(* | \text{The light was})$

get probability
distribution



Как оценить?

- Log-likelihood

$$L(y_{1:M}) = L(y_1, y_2, \dots, y_M) = \sum_{t=1}^M \log_2 p(y_t | y_{<t})$$

Log-likelihood of the text

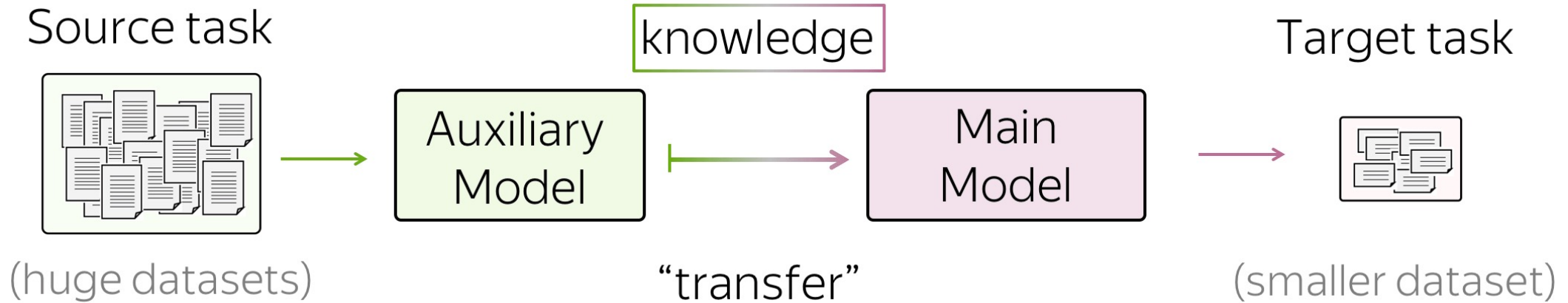
$$Loss(y_{1:M}) = - \sum_{t=1}^M \log p(y_t | y_{<t})$$

Note: cross-entropy (our loss)
is negative log-likelihood

- Perplexity

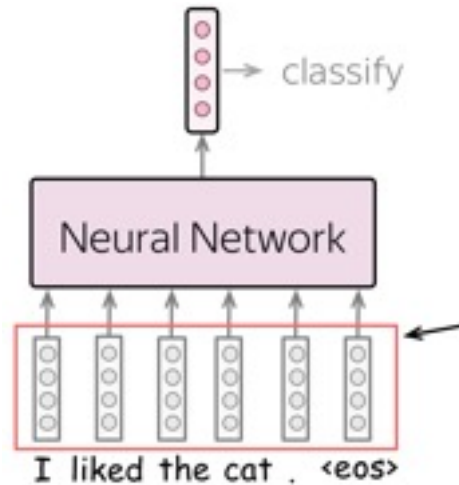
$$Perplexity(y_{1:M}) = 2^{-\frac{1}{M}L(y_{1:M})}$$

Transfer Learning



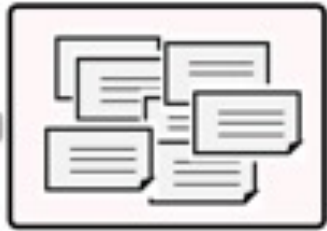
Общая идея – “перенести” знания от одной задачи/модели к другой

Transfer Learning и Word Embeddings



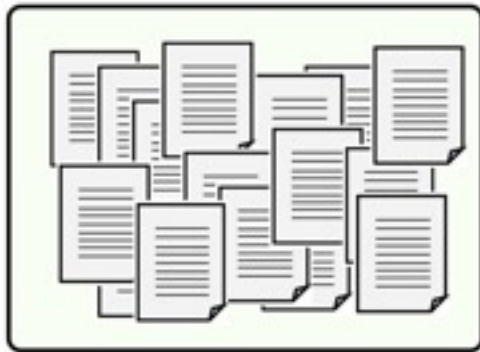
- Обучить с нуля как часть модели
- Использовать фиксированные предобученные
- Инициализировать предобученные и дообучить на своей задаче

Transfer Learning и Word Embeddings



Обучающие данные для текстовой классификации (размеченные):

- небольшой датасет
- специфичный для задачи

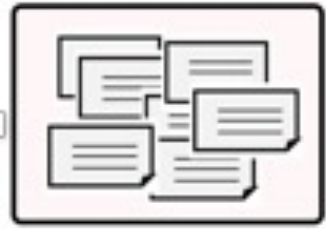


Обучающие данные для векторов слов (неразмеченные):

- большой разнообразный датасет
- задача - общая

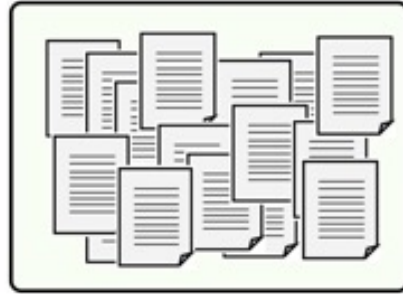
Transfer Learning и Word Embeddings

Обучаем с нуля



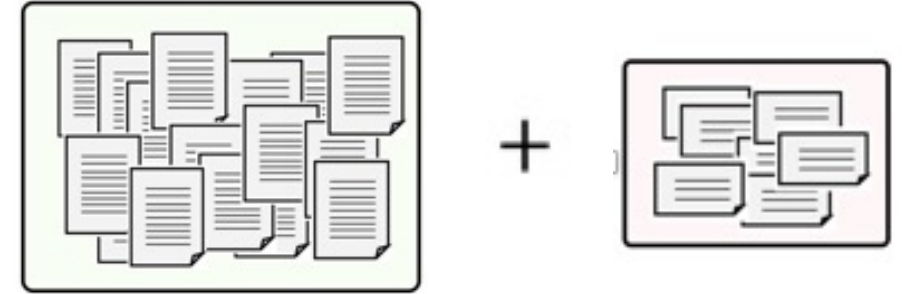
Может быть недостаточно данных для обучения взаимосвязи между словами

Используем предобученные



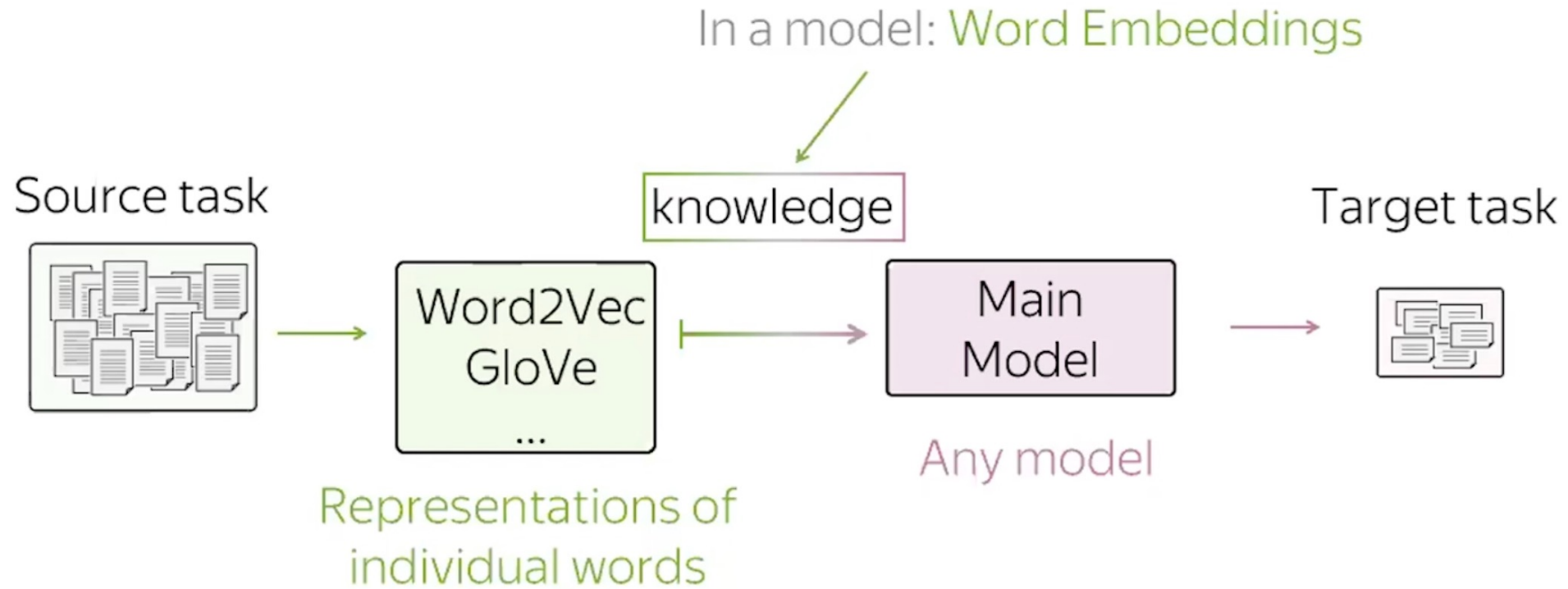
Знаем взаимосвязь между словами, но не специфично к нашей задаче

Инициализируем и дообучиваем



Знаем взаимосвязь между словами и специфику нашей задачи

Transfer Learning и Word Embeddings



Transfer Learning и Word Embeddings

GloVe, Word2Vec → CoVe, ELMo → GPT, BERT

Great idea 1

words



words in context

words in context

What is encoded

How it is used for
downstream tasks

Input for task-
specific models

Input for task-
specific models



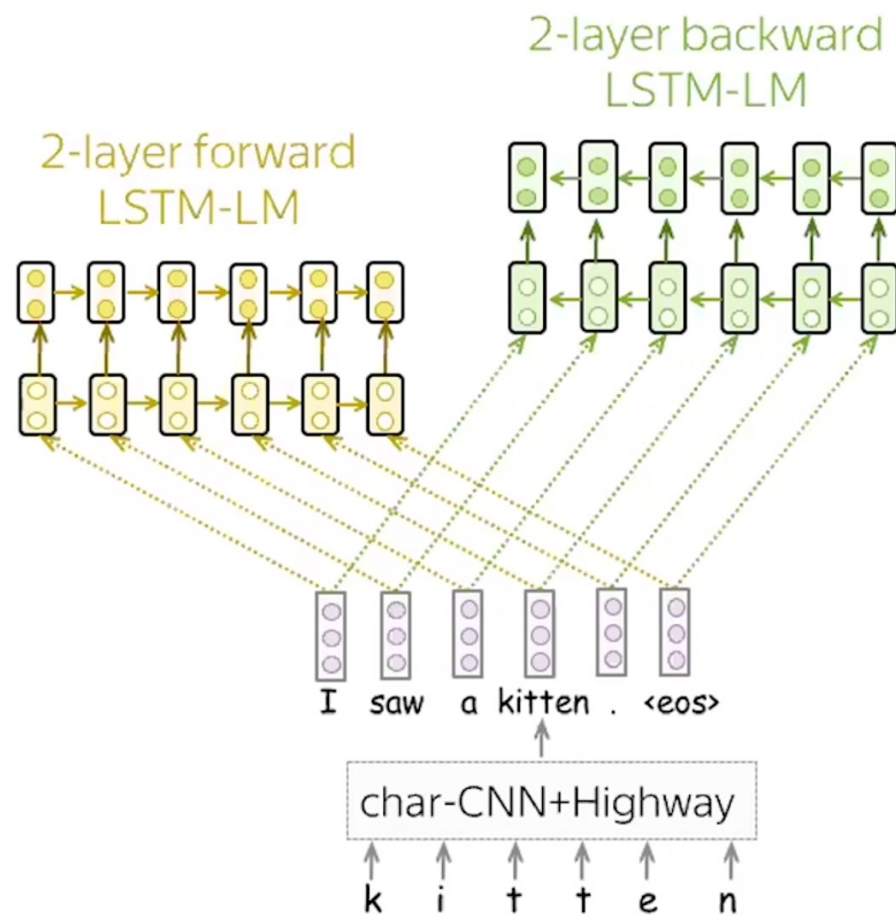
Instead of task-
specific models

Great idea 2

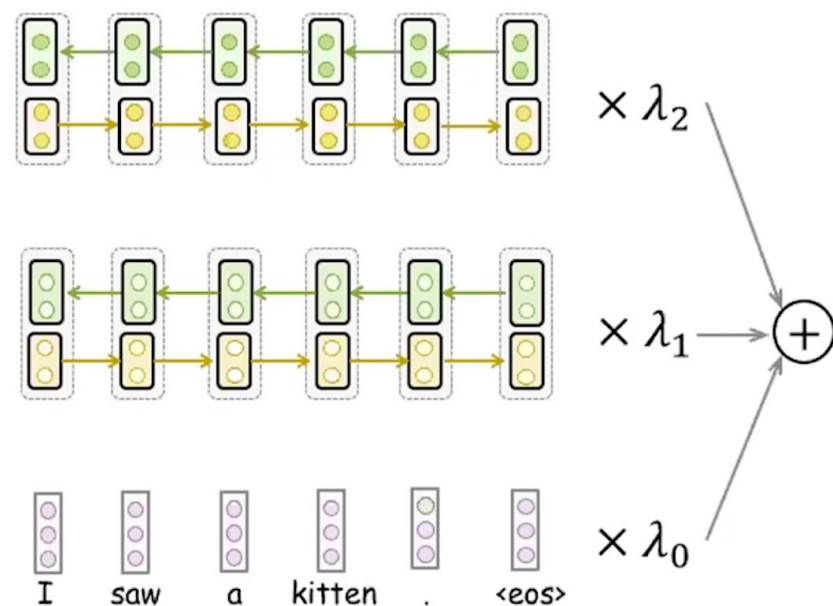
[Ссылка на оригинал картинки](#)

ELMo

от просто слов к словам в контексте

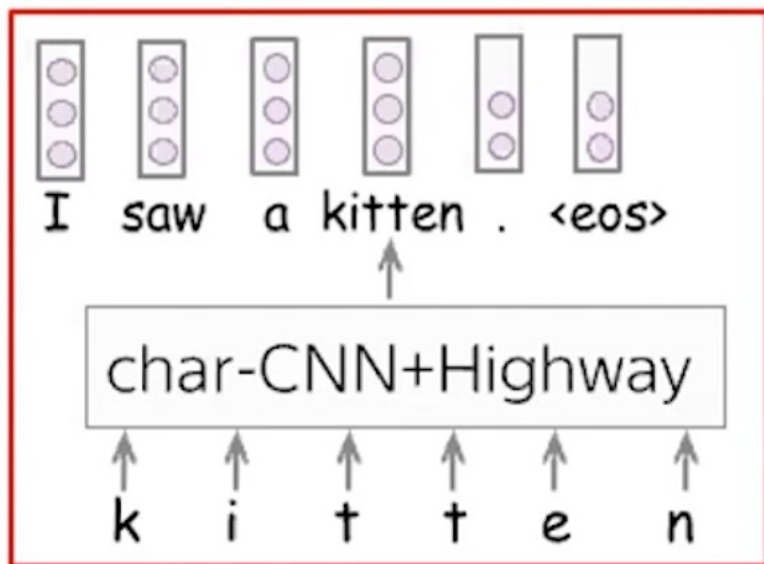


Learn specific $\lambda_0, \lambda_1, \lambda_2$ for each task



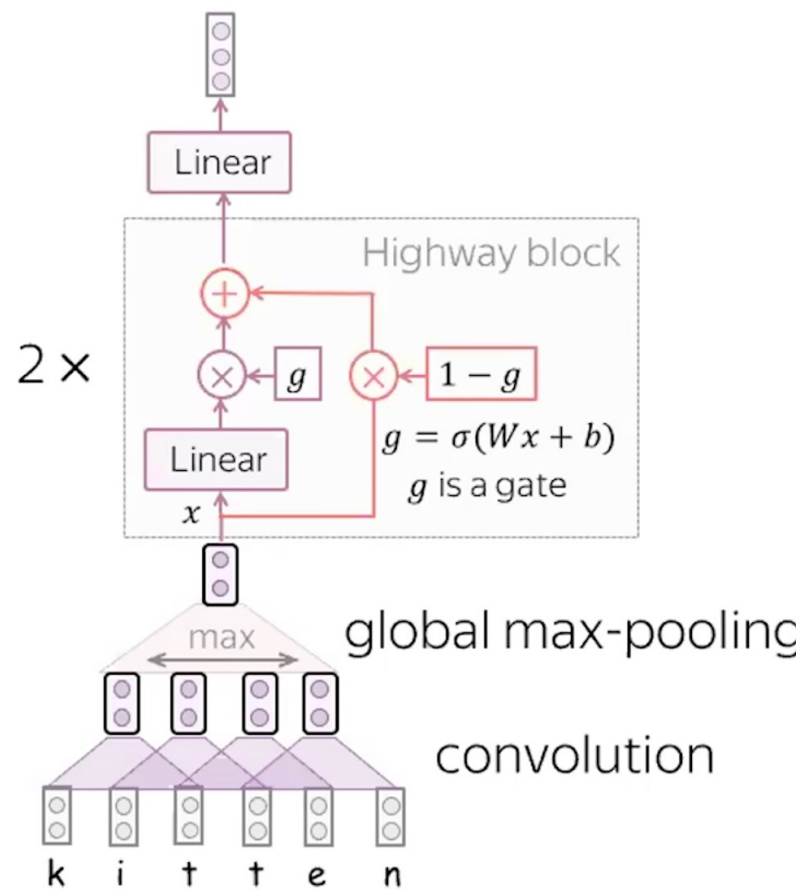
[Ссылка на оригинал картинки](#)

ELMo



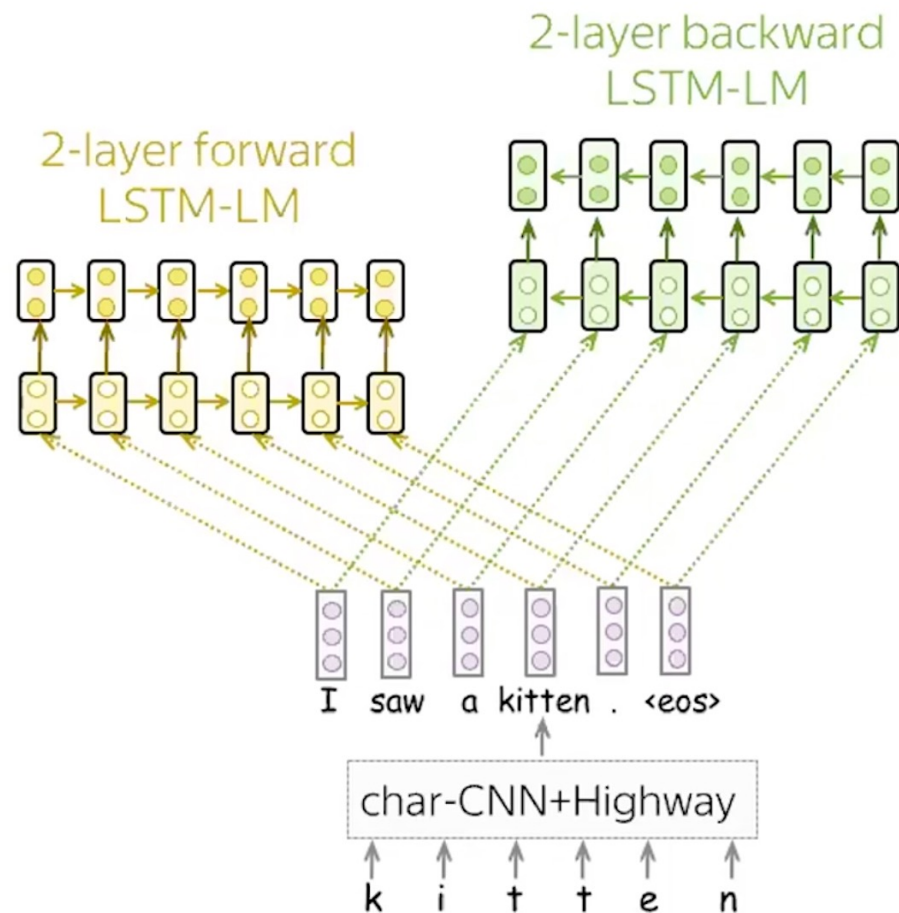
- Initial representation – нейронная сеть на уровне букв
- вектора слов будут знать символы из которых состоят – похожие по написанию слова будут близки, помогает при опечатках
- Out of vocabulary problem – можем использовать эмбеddинг для слов, которые не встречали прежде

ELMo



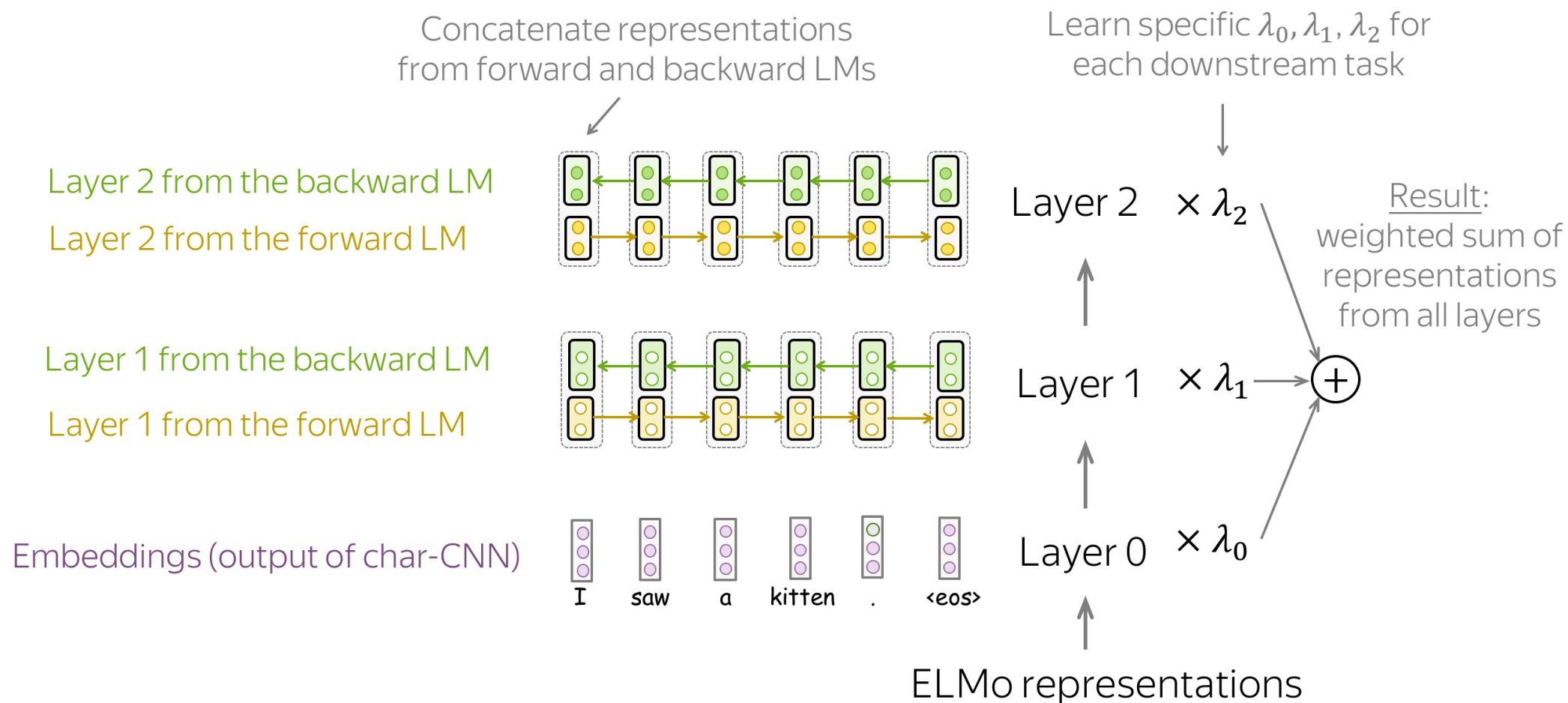
- Initial representation – нейронная сеть на уровне букв
- вектора слов будут знать символы из которых состоят – похожие по написанию слова будут близки, помогает при опечатках
- Out of vocabulary problem – можем использовать эмбединг для слов, которые не встречали прежде

ELMo



- Две двуслойные языковые модели
- Forward модель будет знать левый контекст
- Backward модель будет знать правый контекст

ELMo



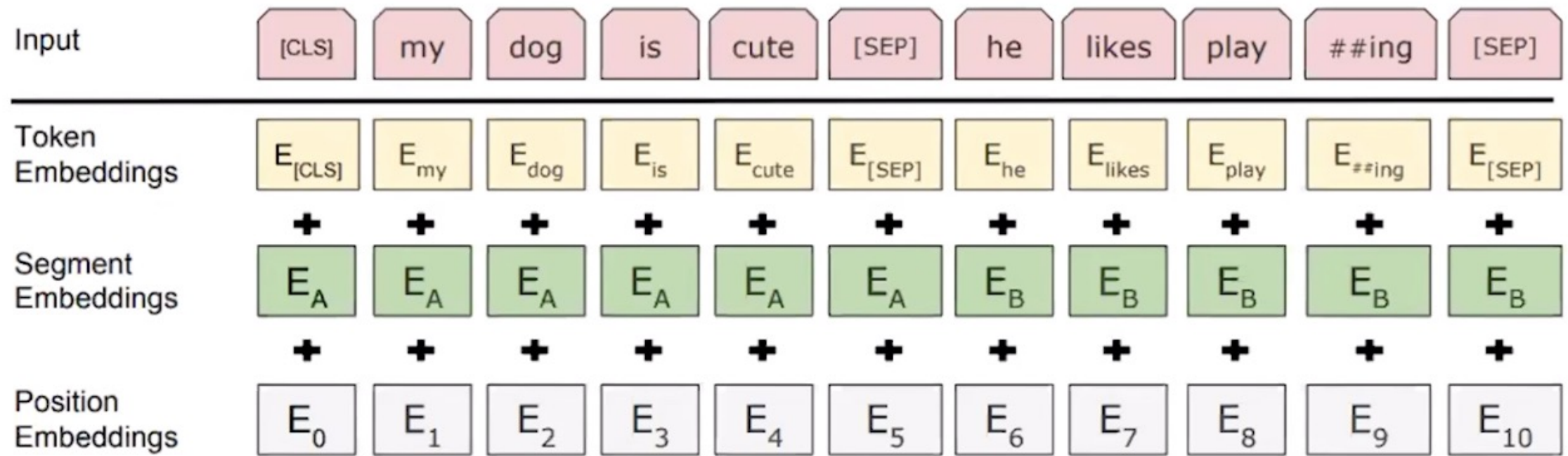
BERT

- До: task-специфичные модели для каждой задачи
- После: единая модель, которая может быть адаптирована для любых задач
- Заменяем не представления слов, а всю модель вместе с представлениями
- Доучиваем модель для конкретной задачи

BERT

- Архитектура – энкодер трансформера (подробнее на след занятии)
- Обучаем – Masked Language modeling и Next Sentence Prediction
- **ОЧЕНЬ** много данных

NSP



NSP

- Передаем последовательности для обучения парами
- 50/50 процентов времени передаем истинное следующее предложение и случайное

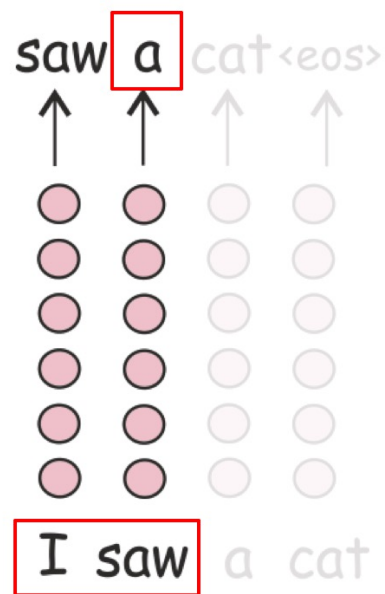
MLM

- 15% слов в каждой последовательности заменяются:
 - токеном [MASK]
 - рандомным токеном
 - исходным токеном
- Затем модель пытается предсказать исходное слово, зная его контекст

MLM

Language Modeling

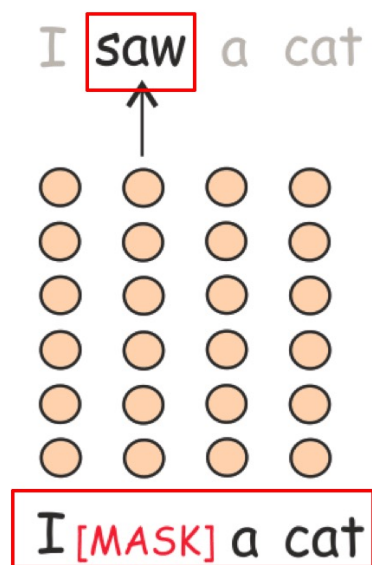
- Target: next token
- Prediction: $P(* | \mathbf{I} \text{ saw})$



left-to-right, does
not see future

Masked Language Modeling

- Target: current token (the true one)
- Prediction: $P(* | \mathbf{I} [\text{MASK}] \text{ a cat})$



sees the whole text, but
something is corrupted

Как использовать?

- Single sentence classification
- Sentence Pair Classification
- Question Answering
- Single sentence tagging

