

**Вложения слов**

Векторы One-Hot

Распределительная семантика

Методы, основанные на подсчете

Word2Vec

Перчатка

Оценка

Анализ и интерпретируемость



Исследовательское мышление



Связанные статьи

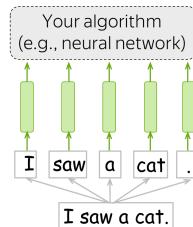


Веселиться!



Вложения слов

То, как модели машинного обучения «видят» данные, отличается от того, как это делаем мы (люди). Например, мы легко понимаем текст «**Я видел кошку**», но наши модели не могут — им нужны векторы признаков. Такие векторы, или вложения слов, представляют собой представления слов, которые можно передать в вашу модель.



Any algorithm for solving a task

Word representation - vector (input for your model/algorithm)

Sequence of tokens

Text (your input)

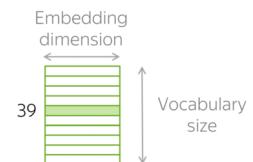
Как это работает: таблица соответствия (словарь)

На практике у вас есть словарь разрешённых слов, который вы выбираете заранее. Для каждого слова из словаря существует таблица соответствия, содержащая его векторное представление. Это представление можно найти по индексу слова в словаре (т.е. вы можете найти векторное представление в таблице, используя индекс слова).

Token index in the vocabulary

39 1592 10 2548 5

I saw a cat .



Для учёта неизвестных слов (тех, которых нет в словаре) словарь обычно содержит специальный токен UNK. В качестве альтернативы, неизвестные токены можно игнорировать или присвоить им нулевой вектор.

I saw a UNK .

I saw a &%! .

not in the vocabulary

Главный вопрос этой лекции: как нам получить эти векторы слов?

Представление в виде дискретных символов: односторонние векторы

Проще всего представить слова в виде векторов типа «one-hot»: для i -го слова в словаре вектор имеет значение 1 по i -му измерению и 0 по остальным. В машинном обучении это самый простой способ представления категориальных признаков.

One is 1, the rest are 0

dog [0...0...010...0...0]

cat [0...010...0...0...0]

table [0...0...0...0010...]

Вы, вероятно, догадываетесь, почему унитарные векторы — не лучший способ представления слов. Одна из проблем заключается в том, что для больших словарей эти векторы будут очень длинными: размерность вектора равна размеру словаря. На практике это нежелательно, но эта проблема не самая критичная.

embedding dimension = vocabulary size

Что действительно важно, так это то, что эти векторы ничего не знают о словах, которые они представляют. Например, векторы с одним значением «думают», что **слово «cat»** так же близко к **слову «dog»**, как и к **слову «table»**! Можно сказать, что векторы с одним значением не передают смысла.

Но как мы узнаем, что такое смысл?

Распределительная семантика

Чтобы уловить значение слов в их векторах, нам сначала необходимо определить понятие значения, которое можно использовать на практике. Для этого попробуем понять, как мы, люди, узнаём, какие слова имеют схожее значение.





Вложения слов

Векторы One-Hot

Распределительная семантика

Методы, основанные на подсчете

Word2Vec

Перчатка

Оценка

Анализ и интерпретируемость



Исследовательское мышление



Связанные статьи



Веселиться!



Как просматривайте слайды в удобном для вас темпе. Постарайтесь заметить, как работает ваш мозг.

Do you know what the word **tezgüino** means ?

(We hope you do not)



- 1.
- 2.
- 3.
- 4.
- 5.
- 6.
- 7.
- 8.
- 9.

Лена : Этот пример взят из заметок Яакоба Эйзенштейна по НЛП; пример с **тезгино** первоначально появился в книге Лина, 1998 .

Увидев, как незнакомое слово используется в разных контекстах, вы смогли понять его значение. Как вам это удалось?

Гипотеза заключается в том, что ваш мозг искал другие слова, которые можно использовать в том же контексте, нашёл несколько (например, «**вино**») и пришёл к выводу, что **слово «tezgüino»** имеет значение, схожее с этими словами. Это гипотеза распределения:

Слова, которые часто встречаются в схожих контекстах, имеют схожее значение .

Лена: Часто можно встретить такую формулировку: «Слово узнаёшь по тому, с кем оно дружит» со ссылкой на Дж. Р. Фёрта в 1957 году, но на самом деле за этим стояло гораздо больше людей, и гораздо раньше. Например, Харрис, 1954 год.

Это чрезвычайно ценная идея: её можно использовать на практике, чтобы заставить векторы слов отражать их значение. Согласно гипотезе распределения, «отражать значение» и «отражать контексты» по сути одно и то же. Следовательно, всё, что нам нужно сделать, — это включить информацию о контекстах слов в словесное представление.

Основная идея : нам нужно включить информацию о контекстах слов в их представление.

Все, чем мы займемся на этой лекции, — это рассмотрим различные способы сделать это.

Методы, основанные на подсчете



Вложения слов

Векторы One-Hot

Распределительная семантика

Методы, основанные на подсчете

Word2Vec

Перчатка

Оценка

Анализ и интерпретируемость



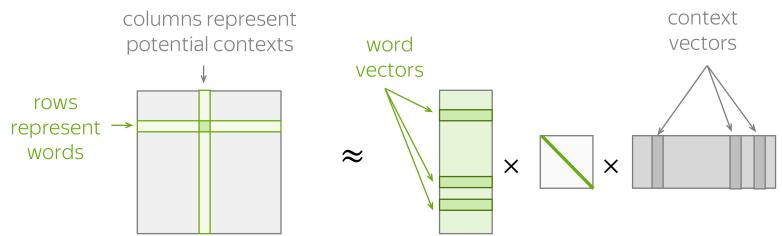
Исследовательское мышление



Связанные статьи



Веселиться!

Reduce dimensionality:
Truncated Singular Value Decomposition (SVD)

Давайте вспомним нашу основную идею:

Основная идея: нам необходимо поместить информацию о контекстах в векторы слов.

Методы, основанные на подсчете, понимают эту идею совершенно буквально:

Как: Введите эту информацию вручную на основе глобальной статистики корпуса.

Общая процедура проиллюстрирована выше и состоит из двух этапов: (1) построение матрицы контекста слова, (2) уменьшение её размерности. Уменьшение размерности необходимо по двум причинам. Во-первых, исходная матрица очень большая. Во-вторых, поскольку многие слова встречаются лишь в ограниченном числе возможных контекстов, эта матрица потенциально содержит много неинформативных элементов (например, нулей).

Чтобы оценить сходство между словами/контекстами, обычно необходимо оценить скалярное произведение нормализованных векторов слов/контекстов (т. е. косинусное сходство).

Чтобы определить метод, основанный на подсчете, нам нужно определить две вещи:

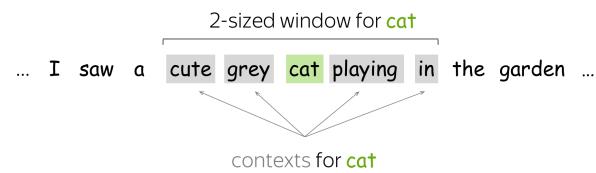
- возможные контексты (включая то, что означает появление слова в контексте),
- понятие ассоциации, т. е. формулы для вычисления элементов матрицы.

Need to define:

- what is context
- how to compute matrix elements

Ниже мы приводим несколько популярных способов сделать это.

Просто: совпадение имеет значение



Самый простой подход — определить контексты как каждое слово в окне размера L. Элемент матрицы для пары слово-контекст (w, c) — это количество появлений w в контексте c . Это самый простой (и очень, очень старый) метод получения векторных представлений.



(Некогда) знаменитая модель HAL (1996) также является модификацией этого подхода. Подробнее об этом упражнении можно узнать в разделе «Исследовательское мышление».

Положительная точечная взаимная информация (PPMI)

Здесь контексты определяются так же, как и раньше, но мера связи между словом и контекстом более продумана: положительный PMI (или сокращённо PPMI). Показатель PPMI широко считается передовым для моделей донейронного распределения сходства.

**Вложения слов**

Векторы One-Hot

Распределительная семантика

Методы, основанные на подсчете

Word2Vec

Перчатка

Оценка

Анализ и интерпретируемость

Исследовательское мышление

Связанные статьи

Веселиться!



Важно: связь с нейронными моделями! Оказывается, некоторые из нейронных методов, которые мы рассмотрим (Word2Vec), неявно аппроксимируют факторизацию (сдвинутой) матрицы PMI. Следите за новостями!

Context:

- surrounding words in a L-sized window

Matrix element:

- $\text{PPMI}(w, c) = \max(0, \text{PMI}(w, c))$, where $\text{PMI}(w, c) = \log \frac{P(w, c)}{P(w)P(c)} = \log \frac{N(w, c) / (w, c)}{N(w)N(c)}$

Латентный семантический анализ (ЛСА): понимание документовЛатентно-семантический анализ (ЛСА)

(ЛСА) анализирует коллекцию документов. В то время как в предыдущих подходах контексты служили только для получения векторов слов и впоследствии отбрасывались, здесь нас также интересует контекст, или, в данном случае, векторы документов. ЛСА — одна из простейших тематических моделей: косинусное сходство векторов документов может использоваться для измерения сходства между документами.

Термин «LSA» иногда относится к более общему подходу применения SVD к матрице термин-документ, где элементы термин-документ могут быть вычислены различными способами (например, простая совместная встречаемость, tf-idf или какое-либо другое взвешивание).

Внимание, анимация! На странице [LSA в Википедии](#) есть прекрасная анимация процесса определения темы в матрице «документ-слово» — взгляните!

Word2Vec: метод, основанный на прогнозировании

Давайте еще раз вспомним нашу основную идею:

Основная идея: нам необходимо поместить информацию о контекстах в векторы слов.

В то время как методы, основанные на подсчете, понимают эту идею совершенно буквально, Word2Vec использует ее по-другому:

Как: Изучите векторы слов, научив их предсказывать контексты.

Word2Vec — это модель, параметрами которой являются векторы слов. Эти параметры итеративно оптимизируются для достижения определённой цели. Эта цель заставляет векторы слов «знать» контексты, в которых может встречаться слово: векторы обучаются предсказывать возможные контексты соответствующих слов. Как вы помните из гипотезы распределения, если векторы «знают» о контекстах, они «знают» и значение слова.

Learned parameters: word vectors

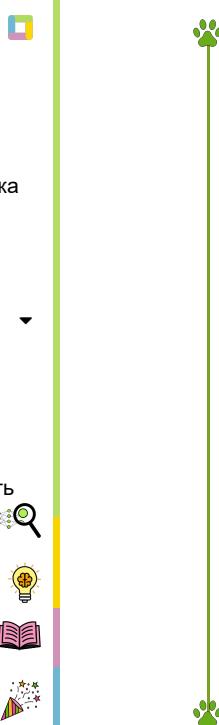
Goal: make each vector “know” about the contexts of its word

How: train vectors to predict possible contexts from words (or, alternatively, words from contexts)

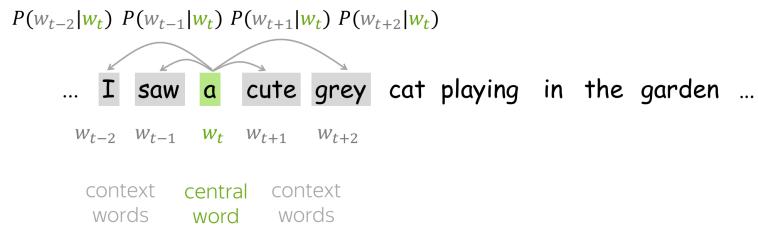
Word2Vec — итеративный метод. Его основная идея заключается в следующем:

- взьмем огромный корпус текстов;
- Пройдитесь по тексту в скользящем окне, перемещаясь по одному слову за раз. На каждом шаге есть центральное слово и контекстные слова (другие слова в этом окне).

- для центрального слова вычислить вероятности контекстных слов;
- скорректируйте векторы, чтобы увеличить эти вероятности.



Как рассмотрите иллюстрацию, чтобы понять основную идею.



context words central word context words

1.

2.

3.

4.

5.

6.

Лена: Идея визуализации взята из курса Стенфордского университета CS224n.

Целевая функция : отрицательный логарифм правдоподобия

Для каждой позиции $t = 1, \dots, T$ в текстовом корпусе Word2Vec предсказывает контекстные слова в пределах окна размером m , учитывая центральное слово w_t :

$$\text{Likelihood} = L(\theta) = \prod_{t=1}^T \prod_{-m \leq j \leq m, j \neq 0} P(w_{t+j}|w_t, \theta),$$

где θ все переменные подлежат оптимизации. Целевая функция (также известная как функция потерь или функция стоимости) $J(\theta)$ — это среднее отрицательное логарифмическое правдоподобие:

$$\text{Loss} = J(\theta) = -\frac{1}{T} \log L(\theta) = -\frac{1}{T} \sum_{t=1}^T \sum_{\substack{-m \leq j \leq m, \\ j \neq 0}} \log P(w_{t+j}|w_t, \theta)$$

agrees with our plan above \mapsto go over text with a sliding window compute probability of the context word given the central

Обратите внимание, насколько хорошо потери согласуются с нашим основным планом, описанным выше: пройтись по тексту в скользящем окне и вычислить вероятности. Теперь давайте разберёмся, как вычислить эти вероятности.

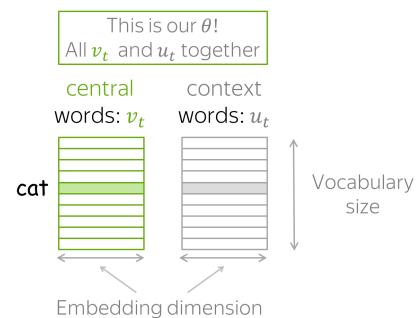
Как рассчитать $P(w_{t+j}|w_t, \theta)$?

Для каждого слова w_t нас будет два вектора:

- v_w когда это центральное слово ;
- u_w когда это контекстное слово .

(После обучения векторов мы обычно отбрасываем векторы контекста и используем только векторы слов.)

Тогда для центрального слова c (— центральный) и контекстное слово o (о — внешнее слово) вероятность контекстного слова равна



Dot product: measures similarity of o and c

$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$

Larger dot product = larger probability
Normalize over entire vocabulary
to get probability distribution**Вложения слов**

Векторы One-Hot

Распределительная семантика

Методы, основанные на подсчете

Word2Vec

Перчатка

Оценка

Анализ и интерпретируемость

Исследовательское мышление

Связанные статьи

Веселиться!

▶ Примечание: это функция softmax ! (нажмите для подробностей)

С этой функцией вам придется довольно часто сталкиваться в ходе изучения НЛП (и в целом в курсе глубокого обучения).

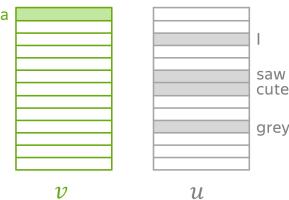


Как просмотрите иллюстрацию. Обратите внимание, что для центральных слов и контекстных слов используются разные векторы. Например, сначала слово a является центральным, и мы используем v_a , но когда это становится контекстом, мы используем u_a вместо.

$$P(u_I|v_a) P(u_{saw}|v_a) P(u_{cute}|v_a) P(u_{grey}|v_a)$$

... I saw a cute grey cat playing in the garden ...

$w_{t-2} w_{t-1} w_t w_{t+1} w_{t+2}$



- 1.
- 2.
- 3.
- 4.
- 5.
- 6.



Как тренироваться : методом градиентного спуска, по одному слову за раз

Напомним, что наши параметры θ являются векторами v_w и u_w для всех слов в словаре. Эти векторы усваиваются путём оптимизации цели обучения с помощью градиентного спуска (с некоторой скоростью обучения). α :

$$\theta^{new} = \theta^{old} - \alpha \nabla_{\theta} J(\theta).$$

По одному слову за раз

Мы выполняем эти обновления по одному за раз: каждое обновление относится к одной паре центрального слова и одного из его контекстных слов. Ещё раз взглянем на функцию потерь:

$$\text{Loss} = J(\theta) = -\frac{1}{T} \log L(\theta) = -\frac{1}{T} \sum_{t=1}^T \sum_{-m \leq j \leq m, j \neq 0} \log P(w_{t+j}|w_t, \theta) = \frac{1}{T} \sum_{t=1}^T \sum_{-m \leq j \leq m, j \neq 0} \log P(w_{t+j}|w_t, \theta)$$

Для центрального слова w_t , убыток содержит отдельный термин $J_{t,j}(\theta) = -\log P(w_{t+j}|w_t, \theta)$ для каждого из его контекстных слов w_{t+j} . Давайте рассмотрим этот термин подробнее и попробуем понять, как выполнить обновление для этого шага. Например, представим, что у нас есть предложение

... I saw a **cute** **grey** **cat** **playing** **in** the garden ...

Курс НЛП | Для вас



Вложения слов

Векторы One-Hot

Распределительная семантика

Методы, основанные на подсчете

Word2Vec

Перчатка

Оценка

Анализ и интерпретируемость



Исследовательское мышление



Связанные статьи



Веселиться!



с центральным словом «**cat**» и четырьмя контекстными словами. Поскольку мы собираемся рассмотреть только один шаг, выберем только одно контекстное слово; например, «**cute**». Тогда член потерь для центрального слова «**cat**» и контекстного слова «**cute**» будет следующим:

$$J_{t,j}(\theta) = -\log P(\text{cute}|\text{cat}) = -\log \frac{\exp u_{\text{cute}}^T v_{\text{cat}}}{\sum_{w \in V_{\text{oc}}} \exp u_w^T v_{\text{cat}}} = -u_{\text{cute}}^T v_{\text{cat}} + \log \sum_{w \in V_{\text{oc}}} \exp u_w^T v_{\text{cat}}.$$

Обратите внимание, какие параметры присутствуют на этом этапе:

- из векторов для **центральных слов**, только v_{cat} ;
- из векторов для контекстных слов, все u_w (для всех слов в словаре).

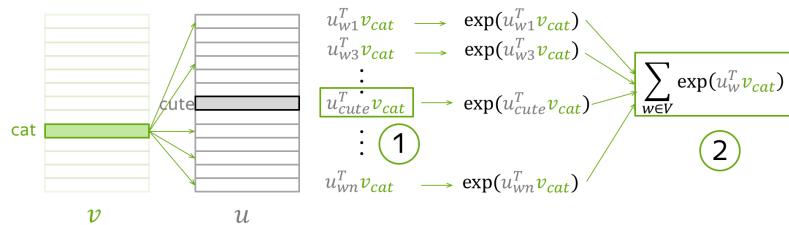
На текущем шаге будут обновлены только эти параметры.

Ниже приведена схематическая иллюстрация выводов для этого шага.

1. Take dot product of v_{cat} with all u

2. exp

3. sum all



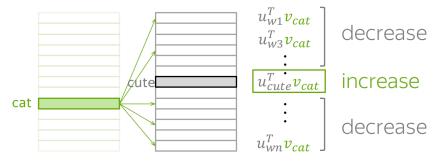
4. get loss (for this one step)

5. evaluate the gradient, make an update

$$J_{t,j}(\theta) = -\underbrace{u_{\text{cute}}^T v_{\text{cat}}}_{1} + \log \underbrace{\sum_{w \in V} \exp(u_w^T v_{\text{cat}})}_{2}$$

$$\begin{aligned} v_{\text{cat}} &:= v_{\text{cat}} - \alpha \frac{\partial J_{t,j}(\theta)}{\partial v_{\text{cat}}} \\ u_w &:= u_w - \alpha \frac{\partial J_{t,j}(\theta)}{\partial u_w} \quad \forall w \in V \end{aligned}$$

Сделав обновление, чтобы минимизировать $J_{t,j}(\theta)$, мы заставляем параметры увеличивать сходство (скалярное произведение) v_{cat} и u_{cute} , и, в то же время, уменьшить сходство между v_{cat} и u_w для всех остальных слов w в словарном запасе.



Это может показаться немного странным: почему мы хотим уменьшить сходство между v_{cat} и все остальные слова, если некоторые из них также являются допустимыми контекстными словами (например, **grey**, **playing**, **in** в нашем примере предложения)?

Но не беспокойтесь: поскольку мы обновляем каждое контекстное слово (и все центральные слова в вашем тексте), в среднем за все обновления наши векторы изучат распределение возможных контекстов.



Попробуйте вывести градиенты на последнем этапе, показанном на иллюстрации выше.

Если вы запутаетесь, можете обратиться к статье [«Word2Vec Parameter Learning Explained»](#).

Более быстрое обучение: отрицательная выборка

**Вложения слов**

Векторы One-Hot

Распределительная семантика

Методы, основанные на подсчете

Word2Vec

Перчатка

Оценка

Анализ и интерпретируемость



Исследовательское мышление



Связанные статьи



Веселиться!



В приведённом выше примере для каждой пары центрального слова и его контекстного слова нам пришлось обновить все векторы контекстных слов. Это крайне неэффективно: время, необходимое для обновления на каждом этапе, пропорционально размеру словаря.

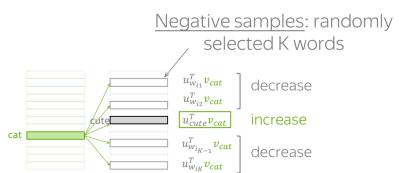
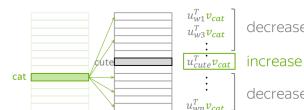
Но почему мы должны учитывать все векторы контекста в словаре на каждом шаге? Например, представьте, что на текущем шаге мы рассматриваем векторы контекста не для всех слов, а только для текущего целевого слова (`cute`) и нескольких случайно выбранных слов. Рисунок иллюстрирует это интуитивно.

Dot product of v_{cat} :

- with u_{cute} - increase,
- with all other u - decrease

Dot product of v_{cat} :

- with u_{cute} - increase,
- with a subset of other u - decrease



Parameters to be updated:

- v_{cat}
- u_w for all w in the vocabulary

Parameters to be updated:

- v_{cat}
- u_{cute} and u_w for w in K negative examples

Как и прежде, мы увеличиваем сходство между v_{cat} и u_{cute} . Разница в том, что теперь мы уменьшаем сходство между v_{cat} и контекстные векторы не для всех слов, а только с подмножеством из K «отрицательных» примеров.

Поскольку у нас большой корпус, в среднем за все обновления мы обновим каждый вектор достаточное количество раз, и векторы по-прежнему смогут достаточно хорошо запоминать взаимосвязи между словами.

Формально новая функция потерь для этого шага имеет вид:

$$J_{t,j}(\theta) = -\log \sigma(u_{cute}^T v_{cat}) - \sum_{w \in \{w_{i_1}, \dots, w_{i_K}\}} \log \sigma(-u_w^T v_{cat}),$$

где w_{i_1}, \dots, w_{i_K} являются K отрицательными примерами, выбранными на этом этапе и $\sigma(x) = \frac{1}{1+e^{-x}}$ — сигмоидальная функция.

Обратите внимание, что $\sigma(-x) = \frac{1}{1+e^x} = \frac{1 \cdot e^{-x}}{(1+e^x) \cdot e^{-x}} = \frac{e^{-x}}{1+e^{-x}} = 1 - \frac{1}{1+e^{-x}} = 1 - \sigma(x)$. Тогда убыток можно записать так:

$$J_{t,j}(\theta) = -\log \sigma(u_{cute}^T v_{cat}) - \sum_{w \in \{w_{i_1}, \dots, w_{i_K}\}} \log(1 - \sigma(u_w^T v_{cat})).$$



Как изменяются градиенты и обновления при использовании отрицательной выборки?

Выбор отрицательных примеров

У каждого слова есть лишь несколько «истинных» контекстов. Поэтому случайно выбранные слова с большой вероятностью окажутся «отрицательными», то есть не имеющими истинного контекста. Эта простая идея используется не только для эффективного обучения Word2Vec, но и во многих других приложениях, некоторые из которых мы рассмотрим далее в курсе.

Word2Vec случайным образом выбирает отрицательные примеры, основываясь на эмпирическом распределении слов. Пусть $U(w)$ быть униграммным распределением слов, т.е. $U(w)$ это частота слова w в корпусе текстов. Word2Vec изменяет это распределение, чтобы чаще выбирать менее часто встречающиеся слова: он выбирает пропорционально $U^{3/4}(w)$.

Варианты Word2Vec: Skip-Gram и CBOW

Существует два варианта Word2Vec: Skip-Gram и CBOW.



Вложения слов

Векторы One-Hot

Распределительная семантика

Методы, основанные на подсчете

Word2Vec

Перчатка

Оценка

Анализ и интерпретируемость



Исследовательское мышление



Связанные статьи



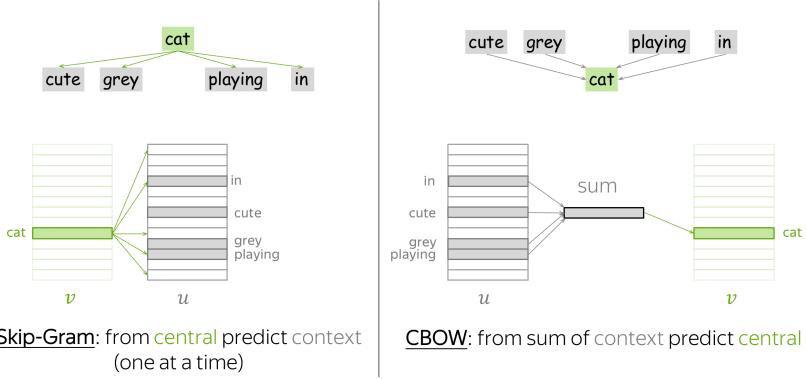
Веселиться!



Skip-Gram — это модель, которую мы рассматривали до сих пор: она предсказывает контекстные слова по центральному слову. Skip-Gram с отрицательной выборкой — самый популярный подход.

Модель CBOW (Continuous Bag-of-Words) предсказывает центральное слово по сумме контекстных векторов. Эта простая сумма векторов слов называется «мешком слов», отсюда и название модели.

... I saw a cute grey cat playing in the garden ...



Как меняются функция потерь и градиенты в модели CBOW?

Если вы запутались, можете снова обратиться к статье «[Word2Vec Parameter Learning Explained](#)».

Дополнительные примечания

Оригинальные статьи Word2Vec:

- Эффективная оценка представлений слов в векторном пространстве
- Распределенные представления слов и фраз и их композиционность

Вы можете ознакомиться с ними, чтобы узнать подробности об экспериментах, реализации и гиперпараметрах. Здесь же мы предоставим вам наиболее важную информацию, которую вам необходимо знать.

Идея не нова

Идея обучения векторов слов (распределённых представлений) не нова. Например, предпринимались попытки обучения векторов слов как части более крупной сети с последующим извлечением слова встраивания. (Подробнее о предыдущих методах можно узнать, например, в кратком изложении в оригинальных статьях Word2Vec).

Что было очень неожиданно в Word2Vec, так это его способность очень быстро изучать высококачественные векторы слов на огромных наборах данных и в больших словарях. И, конечно же, все интересные свойства, которые мы увидим в разделе «[Анализ и интерпретируемость](#)», быстро сделали Word2Vec очень популярным.

Почему два вектора?

Как вы помните, в Word2Vec мы обучаем два вектора для каждого слова: один, когда оно является центральным, и другой, когда оно является контекстным. После обучения контекстные векторы отбрасываются.

Это один из приёмов, благодаря которым Word2Vec стал таким простым. Взгляните ещё раз на функцию потерь (для одного шага):

$$J_{t,j}(\theta) = -u_{\text{cute}}^T v_{\text{cat}} - \log \sum_{w \in V} \exp u_w^T v_{\text{cat}}.$$

Когда центральные и контекстные слова имеют разные векторы, как первый член, так и скалярные произведения внутри экспонент линейны относительно параметров (то же самое относится к отрицательной цели обучения). Поэтому градиенты легко вычисляются.



Повторите выводы (потери и градиенты) для случая с одним вектором для каждого слова ($\forall w \text{ in } V, v_w = u_w$).

Вложения слов

Векторы One-Hot

Распределительная семантика

Методы, основанные на подсчете

Word2Vec

Перчатка

Оценка

Анализ и интерпретируемость



Исследовательское мышление



Связанные статьи



Веселиться!



Хотя стандартной практикой является отбрасывание векторов контекста, было показано, что усреднение векторов слов и контекста может быть более эффективным. [Подробнее здесь.](#)

Лучшее обучение



Есть еще один прием: извлеките больше пользы из этого [упражнения](#) в разделе «[Исследовательское мышление](#)» .

Связь с факторизацией матрицы PMI



Word2Vec SGNS (Skip-Gram с отрицательной выборкой) неявно аппроксимирует факторизацию (сдвинутой) матрицы PMI. [Подробнее здесь.](#)

Влияние размера окна

Размер скользящего окна сильно влияет на результатирующее векторное сходство. Например, в [данной статье](#) отмечается, что окна большего размера, как правило, дают больше тематических сходств (например, слова «собака», «лай» и «поводок» будут сгруппированы вместе, как и слова «ходил», «бежать» и «идти»), в то время как окна меньшего размера, как правило, дают больше функциональных и синтаксических сходств (например, слова «пудель», «питбуль», «ротвейлер» или «идти», «бежать», «приближаться»).

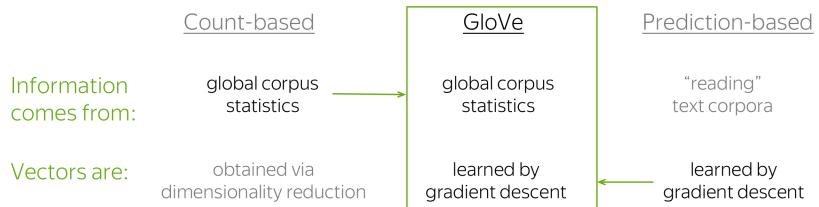
(Несколько) стандартных гиперпараметров

Как всегда, выбор гиперпараметров обычно зависит от решаемой задачи; более подробную информацию можно найти в оригинальных статьях.

Стандартные настройки:

- Модель: Skip-Gram с отрицательной выборкой;
- Количество отрицательных примеров: для небольших наборов данных 15–20; для огромных наборов данных (которые обычно используются) может быть 2–5.
- Размерность встраивания: часто используемое значение — 300, но возможны и другие варианты (например, 100 или 50). Теоретическое объяснение оптимальной размерности см. в разделе «[Связанные статьи](#)» .
- Размер скользящего окна (контекста): 5–10.

GloVe: глобальные векторы для представления слов



Модель **GloVe** представляет собой комбинацию методов, основанных на подсчёте, и методов прогнозирования (например, Word2Vec). Название модели, **GloVe**, расшифровывается как «глобальные векторы» (Global Vectors), что отражает её идею: метод использует глобальную информацию из корпуса для обучения векторов .



Вложения слов

Векторы One-Hot

Распределительная семантика

Методы, основанные на подсчете

Word2Vec

Перчатка

Оценка

Анализ и интерпретируемость



Исследовательское мышление



Связанные статьи



Веселиться!



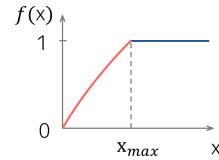
Как мы видели ранее, простейший метод, основанный на подсчёте, использует подсчёт совместного появления для измерения связи между словом w и контекстом $c : N(w, c)$. GloVe также использует эти подсчёты для построения функции потерь:

$$J(\theta) = \sum_{w,c \in V} f(N(w, c)) \cdot (u_c^T v_w + b_c + b_w - \log N(w, c))^2$$

context vector word vector bias terms (also learned)

Weighting function to:

- penalize rare events
- not to over-weight frequent events



$$\begin{cases} (x/x_{max})^\alpha & \text{if } x < x_{max}, \\ 1 & \text{otherwise.} \end{cases}$$

$\alpha = 0.75, x_{max} = 100$

Подобно Word2Vec, у нас также есть разные векторы для центральных и контекстных слов — это наши параметры. Кроме того, метод имеет скалярный член смещения для каждого вектора слов.

Что особенно интересно, так это то, как GloVe контролирует влияние редких и частых слов: потеря для каждой пары (w, c) взвешивается таким образом, что

- редкие события наказываются,
- очень частые события не имеют избыточного веса.

Лена: Функция потерь и так выглядит разумно, но в [оригинальной статье GloVe](#) есть очень хорошая мотивация, приводящая к вышеприведённой формуле. Я не буду её здесь приводить (нужно же когда-нибудь закончить лекцию, верно?..), но вы можете прочитать её сами — она действительно очень интересная!

Оценка вложений слов

Как понять, что один метод получения векторных представлений слов лучше другого? Существует два типа оценки (не только для векторных представлений слов): внутренняя и внешняя.

Внутренняя оценка : на основе внутренних свойств

Этот тип оценки рассматривает внутренние свойства векторных представлений, то есть насколько хорошо они передают смысл. В частности, в разделе «[Анализ и интерпретируемость](#)» мы подробно обсудим, как оценивать векторные представления с точки зрения сходства слов и аналогий слов.

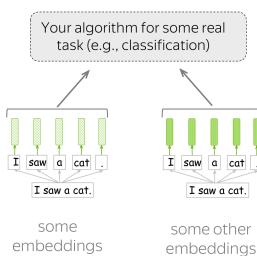
How well do embeddings capture meaning?



- word similarity
- word analogy
- ...

Внешняя оценка : на реальной задаче

Этот тип оценки позволяет определить, какие эмбеддинги лучше подходят для конкретной задачи (например, классификации текста, разрешения кореферентов и т. д.).



Model with which embeddings performs better?

Train the same model several times: one model for each embedding set

For the same dataset, you can get representations using different word embeddings

В этом случае вам придётся несколько раз обучить модель/алгоритм для реальной задачи: по одной модели для каждого оцениваемого эмбеддинга. Затем оцените качество этих моделей, чтобы решить, какие эмбеддинги лучше.

Как выбрать?

Вам придется привыкнуть к тому, что не существует идеального решения и правильного ответа на все ситуации: все всегда зависит от многих факторов.

Intrinsic:

- usually fast
- does not tell what is better in practice

Extrinsic:

- tells directly what is better in practice
- training several real-task models is expensive

**Вложения слов**

Векторы One-Hot

Распределительная семантика

Методы, основанные на подсчете

Word2Vec

Перчатка

Оценка

Анализ и интерпретируемость



Исследовательское мышление



Связанные статьи



Веселиться!



Что касается оценки, вас обычно интересует качество решаемой задачи. Поэтому вас, вероятно, больше заинтересует внешняя оценка. Однако обучение моделей, основанных на реальных задачах, обычно требует много времени и ресурсов, а обучение нескольких из них может оказаться слишком дорогим.

В конце концов, это ваше решение :)



Анализ и интерпретируемость

Лена: Для векторных представлений слов большая часть содержания этой части обычно рассматривается как оценка (внутренняя оценка). Однако, поскольку анализ обычно направлен на изучение того, чему научилась модель (помимо метрик, специфичных для конкретной задачи), я считаю, что её можно представить здесь, в разделе анализа.

Прогуляйтесь по космосу... Семантическому космосу!

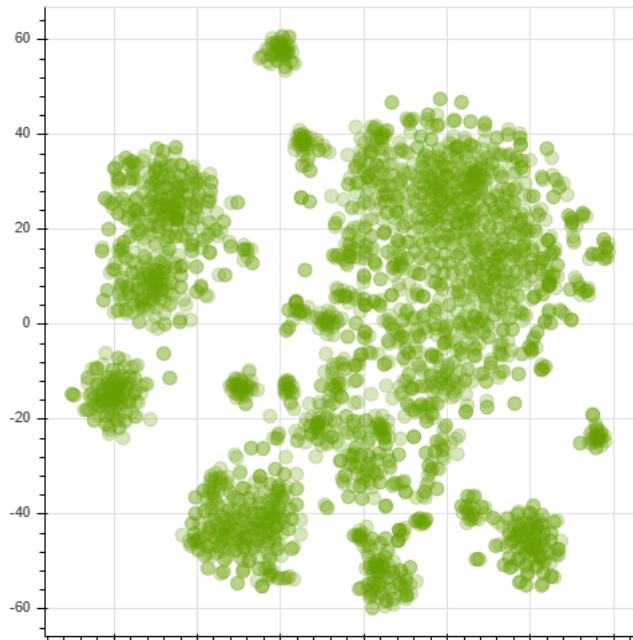
Семантические пространства направлены на создание представлений естественного языка, передающих смысл. Можно сказать, что (хорошие) векторы слов формируют семантическое пространство, и будем называть набор векторов слов в многомерном пространстве «семантическим пространством».

Ниже показано семантическое пространство, сформированное векторами GloVe, обученными на данных Twitter (взятых из [gensim](#)). Векторы были спроектированы в двумерное пространство с помощью t-SNE; здесь представлены только первые 3000 наиболее часто встречающихся слов.



Как: Пройдитесь по семантическому пространству и попытайтесь найти:

- Языковые кластеры: испанский, арабский, русский, английский. Можете ли вы найти ещё языки?
- Кластеры по таким темам, как еда, семья, имена, географические местоположения. Что ещё можно найти?



Ближайшие соседи

Прогуливаясь по семантическому пространству, вы, вероятно, заметили, что точки (векторы), расположенные рядом, обычно имеют близкие значения. Иногда даже редкие слова распознаются очень хорошо. Взгляните на пример: модель поняла, что такие слова, как *leptodactylidae* или *litoria*, близки к слову *frog*.



Вложения слов

Векторы One-Hot

Распределительная семантика

Методы, основанные на подсчете

Word2Vec

Перчатка

Оценка

Анализ и интерпретируемость



Исследовательское мышление



Связанные статьи



Веселиться!



Тесты на схожесть слов	word pair	score	Слово в приг.	Фото
«Оценка»	vulgarism profanity	9.62	frogs	
близайших	subdividing separate	8.67	toad	
соседей	friendships brotherhood	7.5	litoria	
(по	exceedance probability	5.0	leptodactylidae	
	assigned allow	3.5	rana	
	marginalize interact	2.5	lizard	
	misleading beat	1.25	eleutherodactylus	
	radiators beginning	0		

Пример взят со страницы проекта GloVe .

Несколько пар из теста схожести

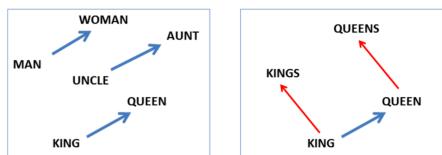
Rare Words .

косинусному сходству или евклидову расстоянию) — один из методов оценки качества полученных векторных представлений. Существует несколько наборов для оценки сходства слов (тестовых наборов). Они состоят из пар слов с оценкой сходства, полученной на основе человеческих суждений. Качество векторных представлений оценивается как корреляция между двумя оценками сходства (модельной и человеческой).

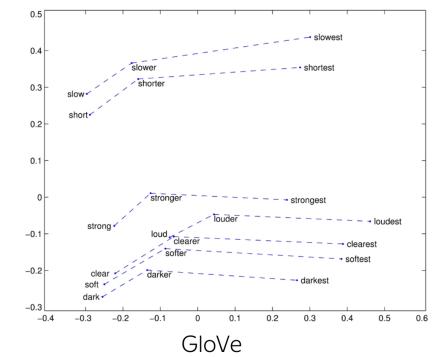
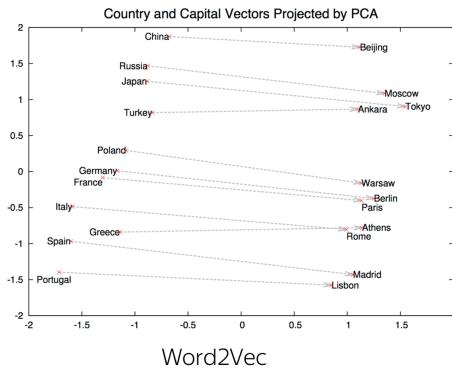
Линейная структура

Хотя результаты по сходству обнадёживают, они не удивительны: в целом, вложения были специально обучены отражать сходство слов. Удивительно то, что многие семантические и синтаксические связи между словами (почти) линейны в векторном пространстве слов.

Например, разница между **словом «король» и словом «королева»** (почти) такая же, как между словом «**мужчина**» и словом «**женщина**». Или слово, похожее на слово «**королева**» в том же смысле, в каком слово «**короли**» похоже на слово «**король**», оказывается словом «**королевы**». **Мужчина-женщина** \approx **Пример короля и королевы**, вероятно, самый популярный, но есть также много других соотношений и забавных примеров.

semantic: $v(\text{king}) - v(\text{man}) + v(\text{woman}) \approx v(\text{queen})$ syntactic: $v(\text{kings}) - v(\text{king}) + v(\text{queens}) \approx v(\text{queens})$ 

Ниже приведены примеры связи «страна-столица» и несколько синтаксических связей.



На конференции ICML 2019 было показано, что существует теоретическое объяснение аналогий в Word2Vec. Подробнее здесь. Лена: Эта статья «[Объяснение аналогий: на пути к пониманию векторных представлений слов](#)» Карла Аллена и Тимоти Хоспдейлса из Эдинбургского университета получила почётное упоминание за лучшую работу на конференции ICML 2019 — совершенно заслуженно!

Тесты по аналогии слов

Эти почти линейные отношения вдохновили на создание нового типа оценки: оценки по аналогии со словами.

Analogy: a is to a^* as b is to __

Task: $v(a^*) - v(a) + v(b) \approx ?$ →

- find the closest vector
- check if it corresponds to the correct word

Даны две пары слов для одного и того же отношения, например, **(мужчина, женщина)** и **(король, королева)**. Задача состоит в том, чтобы проверить, можно ли идентифицировать одно из них по остальным словам. В частности, нам нужно проверить, соответствует ли ближайший вектор к **слову «король» — «мужчина» + «женщина»** слову **«королева»**.

relation	word pair 1		word pair 2	
man-woman	brother	sister	grandson	granddaughter
currency	Angola	kwanza	Iran	rial
opposite	possibly	impossibly	ethical	unethical
past tense	walking	walked	swimming	swam
superlative	easy	easiest	lucky	luckiest

Примеры отношений и пар слов из [тестового набора Google Analogy](#).

Векторы One-Hot

Распределительная семантика

Методы, основанные на подсчете

Word2Vec

Перчатка

Оценка

Анализ и интерпретируемость



Исследовательское мышление



Связанные статьи



Веселиться!



В настоящее время существует несколько тестов на аналогии; к ним относятся стандартные тесты (наборы тестов на аналогии [MSR + Google](#)) и [BATS](#) (Большой набор тестов на аналогии).

Сходства между языками

Мы только что видели, что некоторые связи между словами (почти) линейны в пространстве вложений. Но что происходит между языками? Оказывается, связи между семантическими пространствами также (в некоторой степени) линейны: можно линейно сопоставить одно семантическое пространство с другим, так что соответствующие слова в двух языках будут соответствовать друг другу в новом, общем семантическом пространстве.

The recipe for building large dictionaries from small ones

Ingredients:

- corpus in one language (e.g., [English](#))
- corpus in another language (e.g., [Spanish](#))
- very small dictionary
 $\text{cat} \leftrightarrow \text{gato}$
 $\text{cow} \leftrightarrow \text{vaca}$
 $\text{dog} \leftrightarrow \text{perro}$
 $\text{fox} \leftrightarrow \text{zorro}$
...

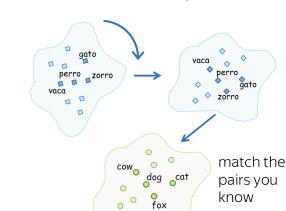
Step 1:

- train embeddings for each language



Step 2:

- linearly map one embeddings to the other to match words from the dictionary



Step 3:

- after matching the two spaces, get new pairs from the new matches

new translations – you learned them!

Рисунок выше иллюстрирует [подход, предложенный Томасом Миколовым и соавторами в 2013 году](#), вскоре после появления оригинального Word2Vec. Формально нам дан набор пар слов и их векторных представлений. $\{x_i, z_i\}_{i=1}^n$, где x_i и z_i — векторы для i -го слова исходного языка и его перевода на целевой язык. Мы хотим найти матрицу преобразования W , такую, что Wz_i приблизительные значения x_i : «соответствует» словам из словаря. Мы выбираем W такой что

$$W = \arg \min_W \sum_{i=1}^n \| Wz_i - x_i \|^2,$$

и изучить эту матрицу методом градиентного спуска.

В оригинальной статье начальный словарный запас состоит из 5 тыс. наиболее употребительных слов с их переводами, а остальное заучивается.



Позже выяснилось, что нам вообще не нужен словарь — мы можем построить отображение между семантическими пространствами, даже ничего не зная о языках! Подробнее здесь.



Действительно ли «истинное» соответствие между языками линейно или оно более сложное? Мы можем проверить это, изучив геометрию изученных семантических пространств. [Подробнее здесь.](#)

Вложения слов

Векторы One-Hot

Распределительная семантика

Методы, основанные на подсчете

Word2Vec

Перчатка

Оценка

Анализ и интерпретируемость



Исследовательское мышление



Связанные статьи



Веселиться!



Идею линейного сопоставления различных наборов вложений для (почти) их соответствия можно использовать и для совершенно другой задачи! Подробнее в разделе «[Исследовательское мышление](#)».



Исследовательское мышление

Как

- Прочтите краткое описание в начале — это наша отправная точка, нечто известное.
- Прочтите вопрос и подумайте: минуту, день, неделю... — дайте себе времени! Даже если вы не думаете об этом постоянно, что-то всё равно может прийти в голову.
- Посмотрите на возможные ответы — предыдущие попытки решить эту задачу. Важно: не обязательно предлагать что-то в точности похожее — помните, каждая статья обычно занимает у авторов несколько месяцев. Важна привычка думать о таких вещах! Всё остальное, что нужно учёному, — это время: пробовать, ошибаться, думать, пока не получится.

Известно, что изучение чего-либо проходит легче, если не сразу получить ответ, а сначала подумать над ним. Даже если вы не хотите становиться исследователем, это всё равно хороший способ учиться!

Методы, основанные на подсчете

Улучшить простые показатели совместного появления

co-occurrence counts
for one sentence

Простейшие подсчёты совместной встречаемости учитывают контекстные слова одинаково, хотя они находятся на разных позициях относительно центрального слова. Например, в одном предложении центральное слово «**cat**» получит подсчёт совместной встречаемости, равный 1 для каждого из слов «**cute**» , «**grey**» , «**playing**» , «**in**» (см. пример справа).

	...I saw a cute grey cat playing in the ...
grey	0 0 1 1 0 1 1 0 0
cat	0 0 0 1 1 0 1 1 0

? Однаково ли важны контекстные слова на разных расстояниях? Если нет, как можно изменить показатели их совместного употребления?

► Возможные ответы

? В языке порядок слов важен; в частности, левый и правый контексты имеют разное значение. Как отличить левый и правый контексты?

► Один из существующих подходов

Word2Vec



Вложения слов

Векторы One-Hot

Распределительная семантика

Методы, основанные на подсчете

Word2Vec

Перчатка

Оценка

Анализ и интерпретируемость



Исследовательское мышление



Связанные статьи



Веселиться!



Все ли контекстные слова одинаково важны для обучения?

В процессе обучения Word2Vec мы обновляем каждое контекстное слово. Например, для центрального слова «**кот**» мы обновляем каждое из слов «**милый**», «**серый**», «**играющий**», «**в**».

...I saw a cute grey cat playing in the ...
...I saw a cute grey cat playing in the ...
...I saw a cute grey cat playing in the ...
...I saw a cute grey cat playing in the ...
...I saw a cute grey cat playing in the ...

? Все ли контекстные слова одинаково важны?

Какие типы слов несут больше/меньше информации, чем другие? Подумайте о некоторых характеристиках слов, которые могут влиять на их важность. Не забудьте предыдущее упражнение!

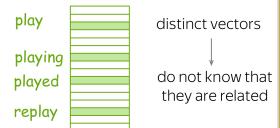
► Возможные ответы

? Как это можно использовать для модификации обучения?

► Хитрости от оригинального Word2Vec

Использовать информацию о подсловах («изобрести» FastText)

Обычно мы используем справочную таблицу, где каждому слову назначается отдельный вектор. По своей конструкции эти векторы не имеют никакого представления о подсловах, из которых они состоят: вся информация, которой они обладают, получена из контекстов.



? Представьте, что у векторных представлений слов есть некое понимание подслов, из которых они состоят. Зачем это может быть полезно?

► Возможные ответы

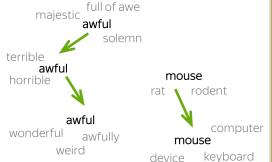
? Как включить информацию о подсловах в вложения? Предположим, что обучающий конвейер фиксирован, например, Skip-Gram с отрицательной выборкой.

► Один из существующих подходов (FastText)

Семантическое изменение

Определите слова, изменившие свое употребление

Представьте, что у вас есть текстовые корпуса из разных источников: временные периоды, группы населения, географические регионы и т. д. В цифровых гуманитарных науках и вычислительных социальных науках люди часто хотят найти слова, которые в этих корпусах используются по-разному.



? Имея два корпуса текстов, как определить, какие слова используются по-разному/имеют разное значение? Не стесняйтесь придумывать самые простые способы!

► Некоторые из существующих попыток



Связанные статьи

Курс НЛП | Для вас



Вложения слов

Векторы One-Hot

Распределительная семантика

Методы, основанные на подсчете

Word2Vec

Перчатка

Оценка

Анализ и интерпретируемость



Исследовательское мышление



Связанные статьи



Веселиться!



Как

- Высокий уровень: просмотрите основные результаты в кратких обзورах — получите представление о том, что происходит в данной области.
- Немного глубже: по темам, которые вас интересуют, читайте более подробные обзоры с иллюстрациями и пояснениями. Просмотрите ход рассуждений авторов и их ключевые наблюдения.
- Подробный анализ: прочтите понравившиеся статьи. Теперь, когда вы поняли основную идею, будет проще!

Что внутри:

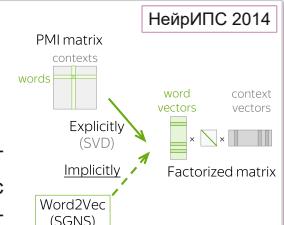
- Старая добрая классика
- Анализ геометрии
- Предубеждения в встраивании слов
- Семантическое изменение
- Теория спешит на помощь! — скоро!
- Кросс-языковые вставки — скоро
- ...будет обновлено

Старая добрая классика

Встраивание нейронных слов как неявная матричная факторизация

Омер Леви, Йоав Голдберг

Теоретически, Word2Vec не так уж сильно отличается от подходов к матричной факторизации! Skip-gram с отрицательной выборкой (SGNS) неявно факторизует сдвинутую поточечно матрицу взаимной информации (PMI): $PMI(w, c) = \log \frac{P(w, c)}{P(w)P(c)}$, где количество отрицательных примеров в отрицательной выборке.

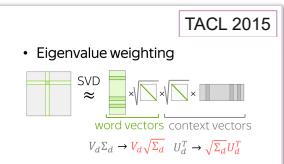


► Подробнее

Улучшение сходства распределения с использованием уроков, извлеченных из векторных представлений слов

Омер Леви, Йоав Голдберг, Идо Даган

В какой-то момент считалось, что встраивание, основанное на прогнозировании, лучше, чем основанное на подсчете. Но это не так: мы можем адаптировать некоторые «трюки» из реализации word2vec к моделям, основанным на подсчете, и добиться тех же результатов. Кроме того, при правильной оценке GloVe хуже Word2Vec.



$$\text{PMI}(w, c) = \log \frac{P(w, c)}{P(w)P(c)} \quad P^*(c) = \frac{N^*(c)}{\sum N^*(c)}$$

• Try $w+c$ for both Word2Vec, Glove

► Подробнее

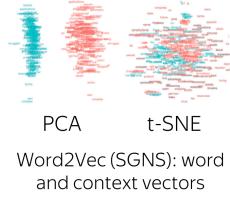
Анализ геометрии



Странная геометрия скрап-грамммы с отрицательной выборкой

Дэвид Мимно, Лора Томпсон

ЭМНЛП 2017



Вложения слов

Векторы One-Hot

Распределительная семантика

Методы, основанные на подсчете

Word2Vec

Перчатка

Оценка

Анализ и интерпретируемость



Исследовательское мышление



Связанные статьи



Веселиться!



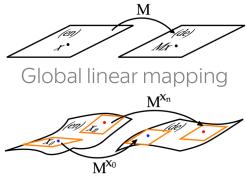
Отрицательная цель выборки влияет на геометрию вложений: векторы слов v_w лежат в узком конусе, диаметрально противоположном векторам контекста u_w . Кроме того, в отличие от GloVe, векторы контекста в Word2Vec направлены в сторону от векторов слов.

► Подробнее

Характеристика отклонений от линейности в переводе слов

Ндапа Накашоле, Рафаэль Флогер

ACL 2018



Мы узнали, что можем (почти) линейно сопоставлять семантические пространства разных языков. Но действительно ли «истинное» базовое сопоставление между языками линейно? Если оно линейно в глобальном масштабе, то все локальные линейные сопоставления должны быть подобны (глобальному линейному сопоставлению, а следовательно, и друг другу). Что ж, это не так.

► Подробнее

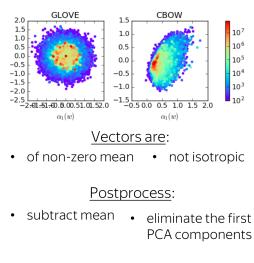
Всё, кроме вершины: простая и эффективная постобработка текстовых представлений

Цзяци Ми, Прамод Вишванатх

МКЭР 2018

Лена: Это пример того, как анализ может улучшить качество!

Авторы заметили, что (i) вложения имеют ненулевое среднее значение и (ii) ранние сингулярные значения значительно больше остальных. Устранив эти свойства, авторы получили значительные улучшения как во внутренней, так и во внешней оценке.



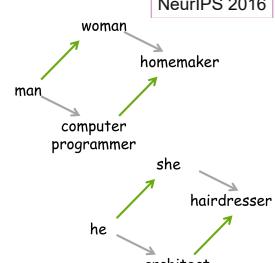
► Подробнее

Предубеждения в встраивании слов

Мужчина для программиста — то же, что женщина для домохозяйки? Устранение искажений вложенных слов

Толга Болукбаси, Кай-Вей Чанг, Джеймс Зоу, Венкатеш Салиграма, Адам Калай

NeurIPS 2016



Вставки слов предвзяты. Например, хотя их аналогия может быть желательной, например, « мужчина для женщины как король для королевы », но также и « мужчина для женщины как врач для медсестры », что является нежелательной ассоциацией.

► Подробнее

**Вложения слов**

Векторы One-Hot

Распределительная семантика

Методы, основанные на подсчете

Word2Vec

Перчатка

Оценка

Анализ и интерпретируемость



Исследовательское мышление



Связанные статьи

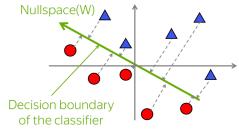


Веселиться!

**Обнулить: защита защищенных атрибутов с помощью итеративной проекции нулевого пространства**

ACL 2020

Шаули Равфогель, Янаи Элазар, Хила Гонен, Майкл Твитон, Йоав Голдберг



Nullspace projection: project to the decision boundary of the classifier



Итеративная проекция нулевого пространства для устранения смещения векторных представлений слов:

- обучить линейный классификатор W предсказать свойство по вложениям (например, пол),
- линейно проецировать вложения на W 's nullspace ($x \rightarrow Px, W(Px) = 0$) - удалить информацию, использованную для прогнозирования;
- повторять до тех пор, пока классификатор не перестанет быть способным что-либо предсказать.

▶ Подробнее

Семантическое изменение

Представьте, что у вас есть корпуса текстов из разных источников: периоды времени, группы населения, географические регионы и т. д. В этой части задача состоит в том, чтобы найти слова, которые используются по-разному в этих корпусах.

Диахронические вложения слов раскрывают статистические законы семантических изменений

ACL 2016

Уильям Л. Хэмилтон, Юре Лесковец, Дэн Джурафски

Лена : Эта статья была использована в разделе «Исследования и размышления». Здесь я скрыла от вас ссылки и содержание — лучше зайдите туда и подумайте. Но если хотите, можете узнать о статье здесь.

Осторожно, спойлер!

I don't want to think, show me the paper!

**Простой, интерпретируемый и стабильный метод обнаружения слов с измененным употреблением в корпусах**

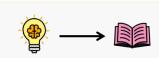
ACL 2020

Хила Гонен, Ганеш Джавахар, Джаме Седда, Йоав Голдберг

Лена : Эта статья была использована в разделе «Исследования и размышления». Здесь я скрыла от вас ссылки и содержание — лучше зайдите туда и подумайте. Но если хотите, можете узнать о статье здесь.

Осторожно, спойлер!

I don't want to think, show me the paper!

**Теория спешит на помощь!****Вскоре:**

- О размерности вложения слов
- Объяснение аналогий: к пониманию вложений слов



Межъязыковые вложения

Курс НЛП | Для вас



Вложения слов

Векторы One-Hot

Распределительная семантика

Методы, основанные на подсчете

Word2Vec

Перчатка

Оценка

Анализ и интерпретируемость



Исследовательское мышление



Связанные статьи



Веселиться!



Вскоре:

- Перевод слов без параллельных данных
- ...будет обновлено



Веселиться!

Серфер семантического пространства

Обычно мы хотим, чтобы словесные представления рассуждали так же, как люди. Но давайте попробуем наоборот: попробуем мыслить как словесные представления.

Вы увидите аналогичный пример, например, король - мужчина + женщина = ?, и несколько возможных ответов. Задача — угадать, что думают вложения слов.

Выполните задание (10 примеров) и получите Сертификат семантического серфера !

Вставки слов: мы использовали glove-twitter-100 из [gensim-data](#).

Большое спасибо [Just Heuristic](#) за помощь с техническими проблемами! Just Heuristic — просто удовольствие!

лев - большой + маленький = ?

собака

обезьяна

Симба

кот



следующий

Выберите свой ответ

Классификация текста

Введение и наборы данных

Общий вид

Классические методы

Нейронные сети

Многоуровневая классификация

Практические советы

Анализ и интерпретируемость

Исследовательское мышление

Связанные статьи

Веселиться!



Классификация текста

Классифицируемый объект: письмо почты с клаудом. Классификация: спам или не спам.

Приложения включают в себя бинарную классификацию документов, классификацию отзывов и т. д.

Две метки, только одна правильная

Классификаторы текста часто используются не как отдельная задача, а

как часть более сложных процессов.

Например, голосовой помощник

классифицирует вашу речь, чтобы понять,

что вы хотите (например, установить

будильник, заказать такси или просто

пообщаться), и передаёт ваше сообщение

различным моделям в зависимости от решения классификатора. Другой пример —

поисковая система: она может использовать классификаторы для определения языка

запроса, прогнозирования типа запроса (например, информационный, навигационный,

транзакционный), понимания того, хотите ли вы просматривать изображения или видео в

дополнение к документам и т. д.

Reviewed 10 papers over the weekend - feel superhuman!

... #fun #science ... #weekend

Многометковая классификация: много меток, некоторые из них могут быть правильными



1.

2.

3.

4.

Поскольку большинство наборов данных классификации предполагают, что верна только одна метка (вы увидите это прямо сейчас!), на лекции мы рассмотрим именно этот тип классификации, то есть классификацию по одной метке. Многометковая классификация будет рассмотрена в отдельном разделе ([Многометковая классификация](#)).

Наборы данных для классификации

Наборы данных для классификации текстов сильно отличаются по размеру (как по размеру самого набора данных, так и по размеру примеров), классифицируемым объектам и количеству меток. Взгляните на статистику ниже.

Набор данных	Тип	Количество этикеток	Размер (обучение/тест)	Средняя длина (токены)
CCT	сентимент	5 или 2	8,5 тыс. / 1,1 тыс.	19
Обзор IMDb	сентимент	2	25к / 25к	271
Обзор Yelp	сентимент	5 или 2	650к / 50к	179
Обзор Amazon	сентимент	5 или 2	3м / 650к	79
ТРЭК	вопрос	6	5,5 тыс. / 0,5 тыс.	10
Ответы Yahoo!	вопрос	10	1,4 млн / 60 тыс.	131
Новости AG	тема	4	120 тыс. / 7,6 тыс.	44
Новости Cogu	тема	6	54к / 6к	737
DBpedia	тема	14	560к / 70к	67

Некоторые наборы данных можно загрузить [здесь](#).

Наиболее популярные наборы данных предназначены для классификации настроений. Они включают обзоры фильмов, мест, ресторанов и продуктов. Существуют также наборы данных для классификации типов вопросов и тем.

Чтобы лучше понять типичные задачи классификации, ниже вы можете рассмотреть примеры из различных наборов данных.



Как выберите набор данных и посмотрите примеры, чтобы понять задачу. Или вернитесь к этому позже!



Классификация текста

Введение и наборы данных

Общий вид

Классические методы

Нейронные сети

Многоуровневая классификация

Практические советы

Анализ и интерпретируемость

Исследовательское мышление

Связанные статьи

Веселиться!

оты и
ку без

е

Выберите набор данных

- CCT Обзор IMDb Обзор Yelp Обзор Amazon
 ТРЭК Ответы Yahoo! Новости AG Новости Cogu
DBpedia

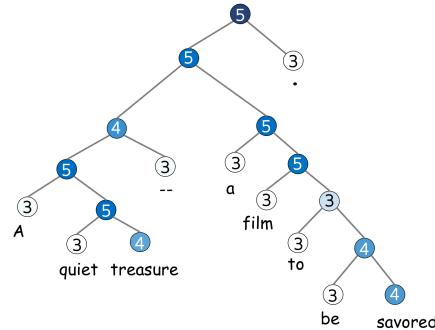
► Описание набора данных (кликните!)

Посмотрите, как из частей складывается смысл предложения.

Метка : 5

Рецензия :

Тихое сокровище — фильм, которым стоит насладиться.



1.
2.
3.

Общий вид

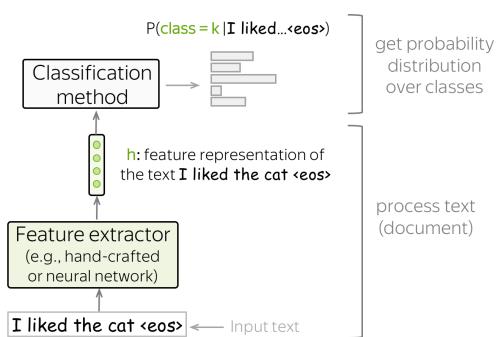
Здесь мы даём общий обзор классификации и вводим обозначения. Этот раздел применим как к классическому, так и к нейронному подходу.

Мы предполагаем, что у нас есть коллекция документов с метками, соответствующими истине. Входными данными классификатора являются документы $x = (x_1, \dots, x_n)$ с токенами (x_1, \dots, x_n) , выход — это метка $y \in 1 \dots k$. Обычно классификатор оценивает распределение вероятностей по классам, и мы хотим, чтобы вероятность правильного класса была максимальной.

Получить представление признаков и классифицировать

Текстовые классификаторы имеют следующую структуру:

- Экстрактор признаков
Экстрактор признаков может быть либо задан вручную (как в [классических подходах](#)), либо обучен (например, с помощью [нейронных сетей](#)).
- Классификатор.
Классификатор должен присваивать вероятности классов данным представлениям признаков текста. Наиболее распространённый способ сделать это — использовать [логистическую регрессию](#), но возможны и другие варианты (например, [наивный байесовский](#) классификатор или [SVM](#)).



В этой лекции мы в основном рассмотрим различные способы построения представления признаков текста и использования этого представления для получения вероятностей

классов.

Генеративные и дискриминационные модели

Курс НЛП | Для вас

Классификация текста

Введение и наборы данных

Общий вид

Классические методы

Нейронные сети

Многоуровневая классификация

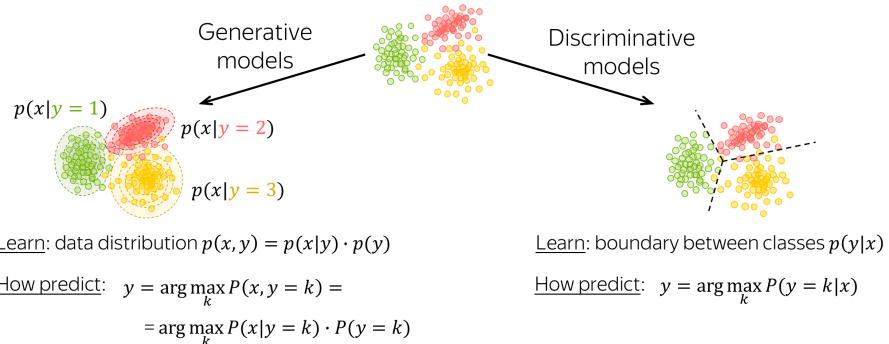
Практические советы

Анализ и интерпретируемость

Исследовательское мышление

Связанные статьи

Веселиться!



Модель классификации может быть либо генеративной , либо дискриминативной .

- Генеративные модели

Генеративные модели изучают совместное распределение вероятностей данных $p(x, y) = p(x|y) \cdot p(y)$. Чтобы сделать прогноз по заданным входным данным x , эти модели выбирают класс с наибольшей вероятностью объединения: $y = \arg \max_k p(x|y = k) \cdot p(y = k)$.

- Дискриминационные модели

Дискриминационные модели интересуются только условной вероятностью $p(y|x)$, то есть они изучают только границу между классами. Чтобы сделать прогноз по входным данным x , эти модели выбирают класс с наибольшей условной вероятностью: $y = \arg \max_k p(y = k|x)$.

В этой лекции мы познакомимся как с генеративными, так и с дискриминативными моделями.

Классические методы классификации текстов

В этой части мы рассмотрим классические подходы к классификации текстов. Они были разработаны задолго до того, как нейронные сети стали популярными, и на небольших наборах данных по-прежнему могут похвастаться производительностью, сравнимой с нейронными моделями.

Лена : Позже в ходе курса мы изучим трансфер обучения , который может улучшить нейронные подходы даже для очень небольших наборов данных. Но давайте разберёмся по пунктам: на данный момент классические подходы — хорошая основа для ваших моделей.

Наивный байесовский классификатор

Ниже представлена общая идея наивного байесовского подхода: мы переписываем условную вероятность класса $P(y = k|x)$ используя правило Байеса и получить $P(x|y = k) \cdot P(y = k)$.

$$y^* = \arg \max_k P(y = k|x) = \arg \max_k \frac{P(x|y = k) \cdot P(y = k)}{P(x)} = \arg \max_k P(x|y = k) \cdot P(y = k)$$

Bayes' rule
(hence Naive Bayes)

Ignore P(x) – it does not influence the argmax

need to define this

Это генеративная модель!

$$y^* = \arg \max_k P(y = k|x) = \arg \max_k P(x|y = k) \cdot P(y = k) = \arg \max_k P(x, y = k)$$

posterior probability:
after looking at data
(i.e., we know x)

prior probability:
before looking at data
(i.e., we don't know x)

joint probability
(hence the model is generative)

$p(x|y = 1)$ $p(x|y = 2)$ $p(x|y = 3)$

Классификация текста

Введение и наборы данных

Общий вид

Классические методы

Нейронные сети

Многоуровневая классификация

Практические советы

Анализ и интерпретируемость

Исследовательское мышление

Связанные статьи

Веселиться!



Наивный байесовский алгоритм является генеративной моделью: он моделирует совместную вероятность данных.

Обратите внимание также на терминологию:

- априорная вероятность $P(y = k)$: вероятность класса до просмотра данных (т.е. до того, как станет известно x);
- апостериорная вероятность $P(y = k|x)$: вероятность класса после просмотра данных (т.е. после знания конкретного x);
- совместная вероятность $P(x, y)$: совместная вероятность данных (т.е. оба примерах этикетки y);
- максимальная апостериорная оценка (MAP): мы выбираем класс с наибольшей апостериорной вероятностью.

Как определить $P(x|y=k)$ и $P(y=k)$?**P(y=k): количество меток**

$P(y = k)$ очень легко получить: мы можем просто оценить долю документов с меткой k (это оценка максимального правдоподобия, MLE). А именно,

$$P(y = k) = \frac{N(y = k)}{\sum_i N(y = i)},$$

где $N(y = k)$ это количество примеров (документов) с меткой k .

P(x|y=k): используйте «наивные» предположения, затем посчитайте

Здесь мы предполагаем, что документ x представлен как набор признаков, например, набор его слов (x_1, \dots, x_n) :

$$P(x|y = k) = P(x_1, \dots, x_n|y = k).$$

Наивные байесовские предположения

- Предположение «мешка слов» : порядок слов не имеет значения,
- Предположение об условной независимости : признаки (слова) независимы относительно данного класса.

Интуитивно мы предполагаем, что вероятность появления каждого слова в документе с классом k не зависит от контекста (ни от порядка слов, ни от других слов). Например, можно сказать, что слова `awesome`, `shiny`, `great` чаще встречаются в документах с положительной тональностью, а слова `terrible`, `bored`, `bad` — в документах с отрицательной тональностью, но мы ничего не знаем о том, как эти (или другие) слова влияют друг на друга.

При таких «наивных» предположениях мы получаем:

$$P(x|y = k) = P(x_1, \dots, x_n|y = k) = \prod_{t=1}^n P(x_t|y = k).$$

Вероятности $P(x_i|y = k)$ оцениваются как доля времени, проведенного с этим словом x_i в документах класса k : среди всех токенов в этих документах:

$$P(x_i|y = k) = \frac{N(x_i, y = k)}{\sum_{t=1}^{|V|} N(x_t, y = k)},$$

где $N(x_i, y = k)$ это количество раз, когда токен x_i появился в документах с меткой k ; $|V|$ — это словарный запас (в более общем смысле — набор всех возможных признаков).

Что, если $N(x_i, y = k) = 0$? Этого нужно избегать!

Что, если $N(x_i, y = k) = 0$, т.е. на тренировке мы не видели жетон x_i в документах с классом k ? Это обнулит вероятность всего документа, а это совсем не то, что нам нужно! Например, если мы не встретили каких-то редких слов (например, «`птеродактиль`» или «`абракадабра`») в обучающих положительных примерах, это не означает, что положительный документ никогда не сможет содержать эти слова.



Классификация текста

Введение и наборы данных

Общий вид

Классические методы

Нейронные сети

Многоуровневая классификация

Практические советы

Анализ и интерпретируемость



Исследовательское мышление



Связанные статьи



Веселиться!



$$N(x_i, y = k) \Rightarrow P(x_i|y = k) = \frac{N(x_i, y = k)}{\sum_{t=1}^{|V|} N(x_t, y = k)} = 0 \Rightarrow P(x|y = k) = \prod_{i=1}^n P(x_i|y = k) = 0$$

In training data, haven't seen token x_i in documents of class k

Чтобы избежать этого, мы воспользуемся простым трюком: мы добавим к количеству всех слов небольшой δ :

$$P(x_i|y = k) = \frac{\delta + N(x_i, y = k)}{\sum_{t=1}^{|V|} (\delta + N(x_t, y = k))} = \frac{\delta + N(x_i, y = k)}{\delta \cdot |V| + \sum_{t=1}^{|V|} N(x_t, y = k)}$$

где δ может быть выбран с использованием перекрестной проверки.

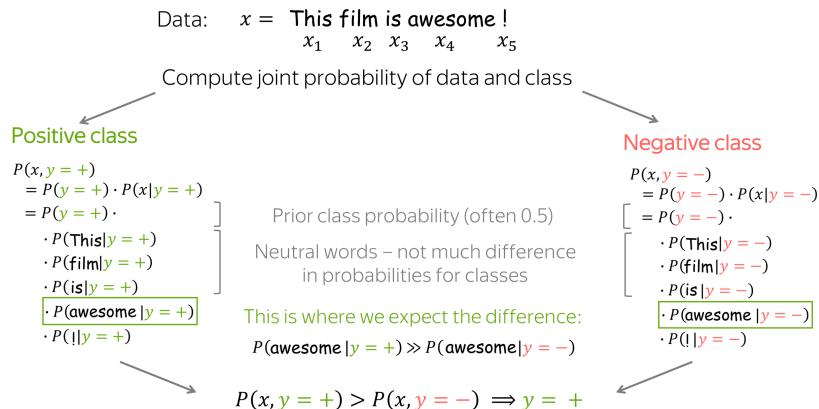
Примечание: это сглаживание Лапласа (также известное как сглаживание Add-1, если $\delta = 1$). Подробнее о сглаживаниях мы поговорим на следующей лекции, когда будем говорить о моделировании языка.

Делая прогноз

Как мы уже упоминали, наивный байесовский алгоритм (и, в более широком смысле, генеративные модели) делает прогноз на основе совместной вероятности данных и класса:

$$y^* = \arg \max_k P(x, y = k) = \arg \max_k P(y = k) \cdot P(x|y = k).$$

Интуитивно наивный байесовский алгоритм предполагает, что некоторые слова служат индикаторами классов. Например, при классификации настроений токены «удивительный» , «блестящий» , «великолепный» будут иметь более высокую вероятность при положительном классе, чем при отрицательном. Аналогично, токены «ужасный» , «скучный» , «плохой» будут иметь более высокую вероятность при отрицательном классе, чем при положительном.



Заключительные замечания по наивному байесовскому анализу

Практическое примечание: сумма логарифмов вероятностей вместо произведения вероятностей

Основное выражение, которое наивный байесовский алгоритм использует для классификации, — это произведение вероятностей:

$$P(x, y = k) = P(y = k) \cdot P(x_1, \dots, x_n|y) = P(y = k) \cdot \prod_{t=1}^n P(x_t|y = k).$$

Произведение многих вероятностей может быть очень нестабильным в численном отношении. Поэтому обычно вместо $P(x, y)$ мы рассматриваем $\log P(x, y)$:

$$\log P(x, y = k) = \log P(y = k) + \sum_{t=1}^n \log P(x_t | y = k).$$

Курс НЛП | Для вас



Классификация текста

Введение и наборы данных

Общий вид

Классические методы

Нейронные сети

Многоуровневая классификация

Практические советы

Анализ и интерпретируемость



Исследовательское мышление



Связанные статьи



Веселиться!



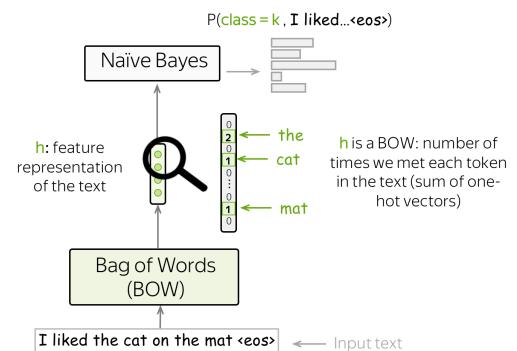
Поскольку нас интересует только argmax, мы можем рассмотреть $\log P(x, y)$ вместо $P(x, y)$.

Важно! Обратите внимание, что на практике мы обычно имеем дело с логарифмическими вероятностями, а не с вероятностями.

Просмотр в общей структуре

Помните наш [общий взгляд](#) на задачу классификации? Мы получаем представление признаков входного текста с помощью какого-либо метода, а затем используем это представление признаков для классификации.

В наивном байесовском алгоритме наши признаки – это слова, а представление признаков – это представление в виде мешка слов (BOW) – суммы однозначных представлений слов. Действительно, для оценки $P(x, y)$ нам нужно только подсчитать, сколько раз каждый токен встречается в тексте.



Дизайн функций

В стандартной настройке в качестве признаков используются слова. Однако вы можете использовать и другие типы признаков: URL, идентификатор пользователя и т. д.



Даже если ваши данные представляют собой простой текст (без сложных элементов вроде URL, идентификатора пользователя и т. д.), вы всё равно можете создавать функции разными способами. Узнайте, как улучшить наивный байесовский алгоритм, в [этом упражнении](#) из раздела «Исследовательское мышление».

Классификатор с максимальной энтропией (он же логистическая регрессия)

В отличие от наивного байесовского классификатора, MaxEnt является дискриминационной моделью, т.е. нас интересует $P(y = k|x)$ и не в совместном распределении $P(x, y)$. Кроме того, мы научимся использовать признаки: это отличается от наивного байесовского алгоритма, где мы сами определили, как использовать признаки.

Здесь нам тоже придётся определять признаки вручную, но у нас больше свободы: признаки не обязательно должны быть категориальными (в наивном байесовском алгоритме они должны быть!). Мы можем использовать представление BOW или придумать что-то более интересное.

Общий конвейер классификации здесь выглядит следующим образом:

- получать $h = (f_1, f_2, \dots, f_n)$ – представление признаков входного текста;
- брать $w^{(i)} = (w_1^{(i)}, \dots, w_n^{(i)})$ – векторы с весами признаков для каждого из классов;
- для каждого класса взвесить признаки, т.е. взять скалярное произведение представлений признаков h с весами признаков $w^{(k)}$:

$$w^{(k)} h = w_1^{(k)} \cdot f_1 + \dots + w_n^{(k)} \cdot f_n, \quad k = 1, \dots, K.$$

Чтобы получить смещение в сумме выше, мы определяем один из признаков как 1 (например, $f_0 = 1$). Затем

$$w^{(k)} h = w_0^{(k)} + w_1^{(k)} \cdot f_1 + \dots + w_n^{(k)} \cdot f_n, \quad k = 1, \dots, K.$$

- получить вероятности классов с помощью softmax:

$$P(class = k | \text{h}) = \frac{\exp(w^{(k)} \cdot \text{h})}{\sum_{i=1}^K \exp(w^{(i)} \cdot \text{h})}.$$



Классификация текста

Введение и наборы данных

Общий вид

Классические методы

Нейронные сети

Многоуровневая классификация

Практические советы

Анализ и интерпретируемость



Исследовательское мышление



Связанные статьи

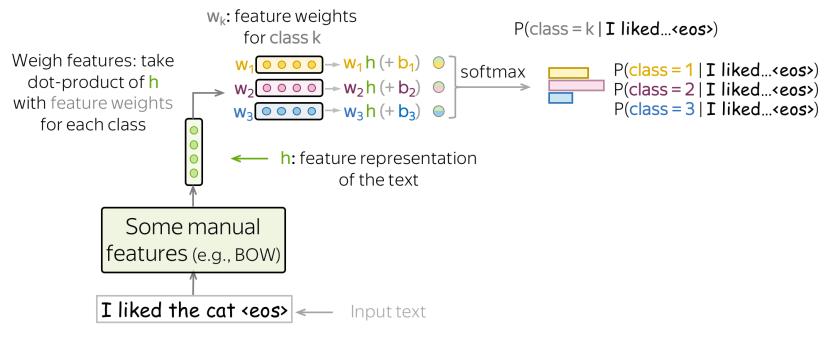


Веселиться!



Softmax нормализует K значения, которые мы получили на предыдущем этапе распределения вероятностей по выходным классам.

Посмотрите на иллюстрацию ниже (классы показаны разными цветами).



Обучение: Оценка максимального правдоподобия

Приведенные примеры обучения x^1, \dots, x^N с этикетками $y^1, \dots, y^N, y^i \in \{1, \dots, K\}$, мы выбираем эти веса $w^{(k)}, k = 1..K$ которые максимизируют вероятность обучающих данных:

$$w^* = \arg \max_w \sum_{i=1}^N \log P(y = y^i | x^i).$$

Другими словами, мы выбираем параметры таким образом, чтобы вероятность появления данных была выше . Поэтому это называется оценкой максимального правдоподобия (ОМП) параметров.

Чтобы найти параметры, максимизирующие логарифмическое правдоподобие данных, мы используем градиентный подъём: постепенно увеличиваем веса в течение нескольких итераций. На каждом шаге мы максимизируем вероятность, которую модель присваивает правильному классу.

Эквивалентность минимизации перекрестной энтропии

Обратите внимание, что максимизация логарифмического правдоподобия данных эквивалентна минимизации перекрестной энтропии между целевым распределением вероятностей $p^* = (0, \dots, 0, 1, 0, \dots)$ (1 для целевой метки, 0 для остальных) и предсказанное моделью распределение $p = (p_1, \dots, p_K), p_i = p(i|x)$:

$$\text{Loss}(p^*, p) = -p^* \log(p) = -\sum_{i=1}^K p_i^* \log(p_i).$$

Поскольку только один из p_i^* не равен нулю (1 для целевой метки, 0 для остальных), мы получим $\text{Loss}(p^*, p) = -\log(p_k) = -\log(p(k|x))$.



Классификация текста

Введение и наборы данных

Общий вид

Классические методы

Нейронные сети

Многоуровневая классификация

Практические советы

Анализ и интерпретируемость



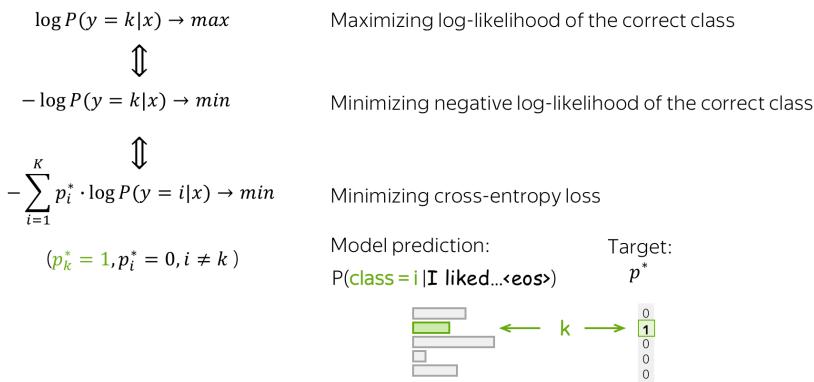
Исследовательское мышление



Связанные статьи



Веселиться!



Эту эквивалентность очень важно понимать: говоря о нейронных подходах, обычно говорят, что они минимизируют потерю кросс-энтропии. Не забывайте, что это то же самое, что максимизировать логарифм правдоподобия данных.

Наивный байесовский анализ против логистической регрессии

Naïve Bayes:

- very simple
- very fast
- interpretable
- assumes that features are conditionally independent
- text representation: manually defined (and often too simple, e.g. BOW)

Logistic Regression:

- quite simple
- interpretable
- does not assume that features are conditionally independent
- not so fast (multiple iterations of gradient ascent)
- text representation: manually defined

Давайте завершим эту часть обсуждением преимуществ и недостатков логистической регрессии и наивного байесовского алгоритма.

• Простота

Оба метода просты; наивный байесовский метод — самый простой.

• Интерпретируемость

Оба метода являются интерпретируемыми: вы можете посмотреть на признаки, которые больше всего повлияли на прогнозы (в наивном байесовском методе — обычно слова, в логистической регрессии — то, что вы определили).

• Скорость обучения.

Наивный байесовский алгоритм обучается очень быстро — для оценки количества импульсов требуется всего один проход по обучающим данным. В случае логистической регрессии это не так: необходимо многократно проходить по данным, пока градиентное восхождение не сойдется.

• Наивный байесовский алгоритм

слишком «наивен» — он предполагает, что признаки (слова) условно независимы для данного класса. Логистическая регрессия не делает такого предположения — можно надеяться, что он лучше.

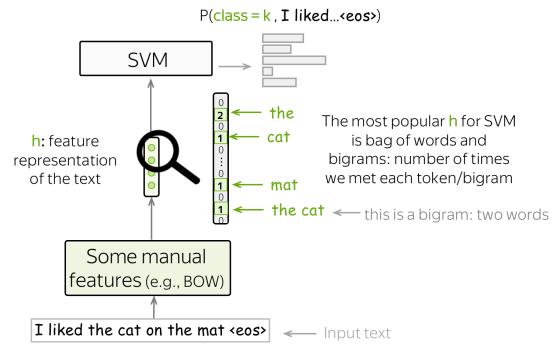
• Текстовое представление: ручное.

Оба метода используют представление признаков, заданных вручную (в наивном байесовском алгоритме стандартным выбором является BOW, но вы всё равно можете выбрать его самостоятельно). Хотя заданные вручную признаки хороши для интерпретируемости, они могут быть не столь эффективны для производительности — вы, вероятно, упустите что-то полезное для задачи.

SVM для классификации текстов

Ещё один метод классификации текстов, основанный на вручную разработанных признаках, — это SVM. Наиболее базовыми (и популярными) признаками для SVM являются «мешок слов» и «мешок n-грамм» (где n-грамма — это кортеж из n слов). С этими простыми признаками SVM с линейным ядром работают лучше, чем наивный

байесовский алгоритм (см., например, статью «[Классификация вопросов с использованием опорных векторов](#)»).

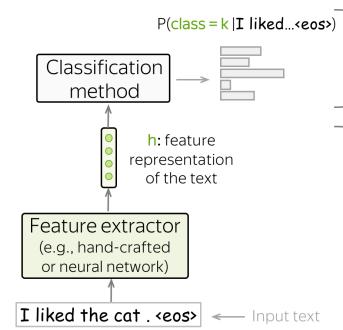


Классификация текста с помощью нейронных сетей

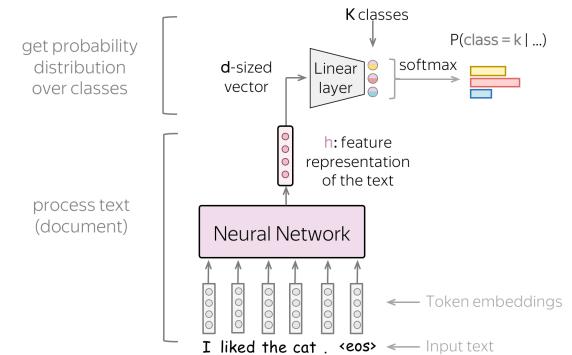
Вместо того, чтобы вручную задавать признаки, позвольте нейронной сети изучать полезные признаки.

Основная идея классификации на основе нейронных сетей заключается в том, что представление признаков входного текста может быть получено с помощью нейронной сети. В этом случае мы передаем вложения входных токенов в нейронную сеть, которая затем формирует векторное представление входного текста. Этот вектор затем используется для классификации.

General Classification Pipeline



Classification with Neural Networks



При работе с нейронными сетями мы можем рассматривать классификационную часть (т. е. как получить вероятности классов из векторного представления текста) очень простым способом.

Векторное представление текста имеет некоторую размерность d , но в конечном итоге нам нужен вектор размером K (вероятности для K классы). Чтобы получить K -размерный вектор из d , мы можем использовать линейный слой. Как только у нас есть K вектор размером с вектор, все, что осталось, это применить операцию softmax для преобразования необработанных чисел в вероятности классов.

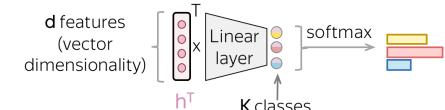
We have:

- h - vector of size d

We need:

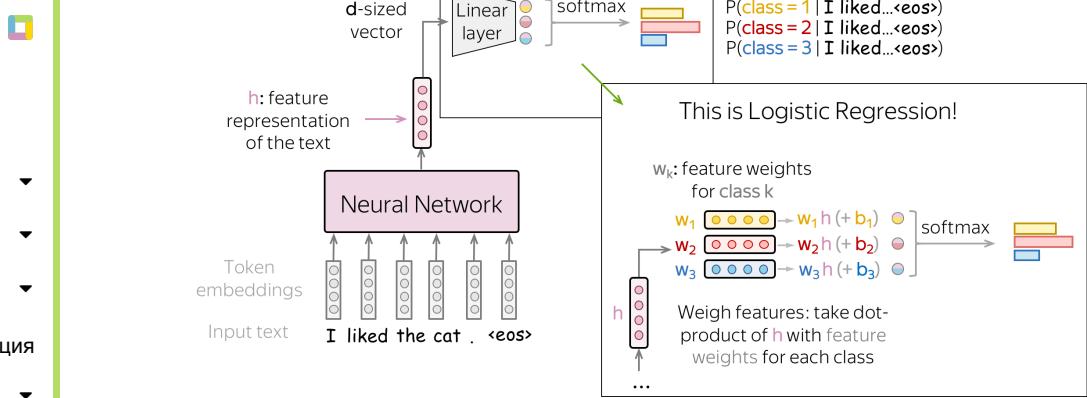
- vector of size K – probabilities for K classes

Transform linearly from size d to size K



Часть классификации: Это логистическая регрессия!

Давайте подробнее рассмотрим нейросетевой классификатор. Мы используем векторное представление входного текста точно так же, как и в логистической регрессии: взвешиваем признаки в соответствии с весами признаков для каждого класса. Единственное отличие от логистической регрессии заключается в том, откуда берутся признаки: они либо задаются вручную (как мы делали раньше), либо получаются нейронной сетью.



Классификация текста

Введение и наборы данных

Общий вид

Классические методы

Нейронные сети

Многоуровневая классификация

Практические советы

Анализ и интерпретируемость

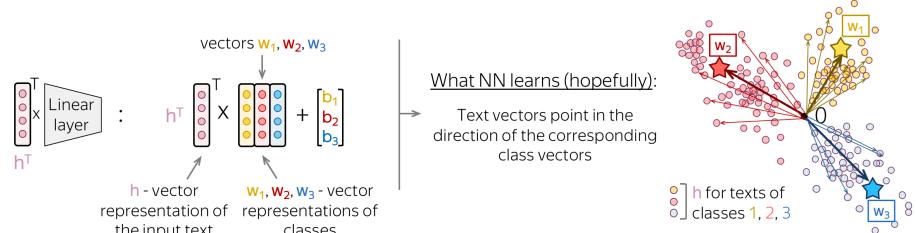


Исследовательское мышление

Связанные статьи

Веселиться!

Интуиция: представление текста указывает на представление класса



Если мы посмотрим на этот последний линейный слой более внимательно, то увидим, что столбцы его матрицы — это векторы w_i . Эти векторы можно рассматривать как векторные представления классов. Хорошая нейронная сеть научится представлять входные тексты таким образом, чтобы текстовые векторы указывали направление соответствующих векторов классов.

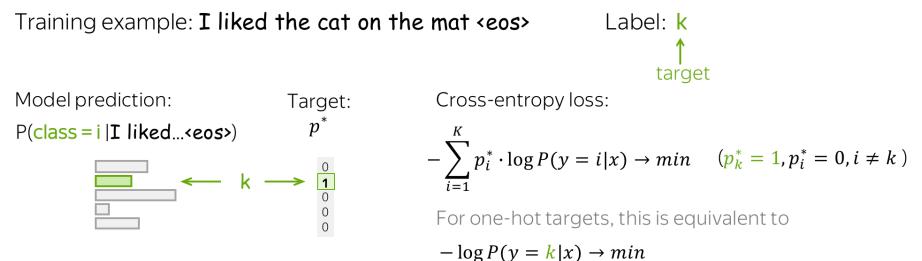
Обучение и потеря кросс-энтропии

Нейронные классификаторы обучены предсказывать распределение вероятностей по классам. Интуитивно понятно, что на каждом шаге мы максимизируем вероятность, которую модель присваивает правильному классу.

Стандартная функция потерь — это кросс-энтропийная потеря . Кросс-энтропийная потеря для целевого распределения вероятностей $p^* = (0, \dots, 0, 1, 0, \dots)$ (1 для целевой метки, 0 для остальных) и предсказанное моделью распределение $p = (p_1, \dots, p_K)$, $p_i = p(i|x)$:

$$\text{Loss}(p^*, p) = -p^* \log(p) = -\sum_{i=1}^K p_i^* \log(p_i).$$

Поскольку только один из p_i^* не равен нулю (1 для целевой метки, 0 для остальных), мы получим $\text{Loss}(p^*, p) = -\log(p_k) = -\log(p(k|x))$. На иллюстрации представлен один из примеров обучения.



В процессе обучения мы постепенно улучшаем веса модели в ходе нескольких итераций по данным: мы итерируемся по обучающим примерам (или группам примеров) и обновляем градиент. На каждом шаге мы максимизируем вероятность, которую модель отнесёт к правильному классу. Одновременно мы минимизируем сумму вероятностей неверных классов: поскольку сумма всех вероятностей постоянна, увеличивая одну вероятность, мы уменьшаем сумму всех остальных (Лена : Здесь я обычно представляю себе стаю котят, которые едят из одной миски: один котёнок всегда ест за счёт других).

Классификация текста

Введение и наборы данных

Общий вид

Классические методы

Нейронные сети

Многоуровневая классификация

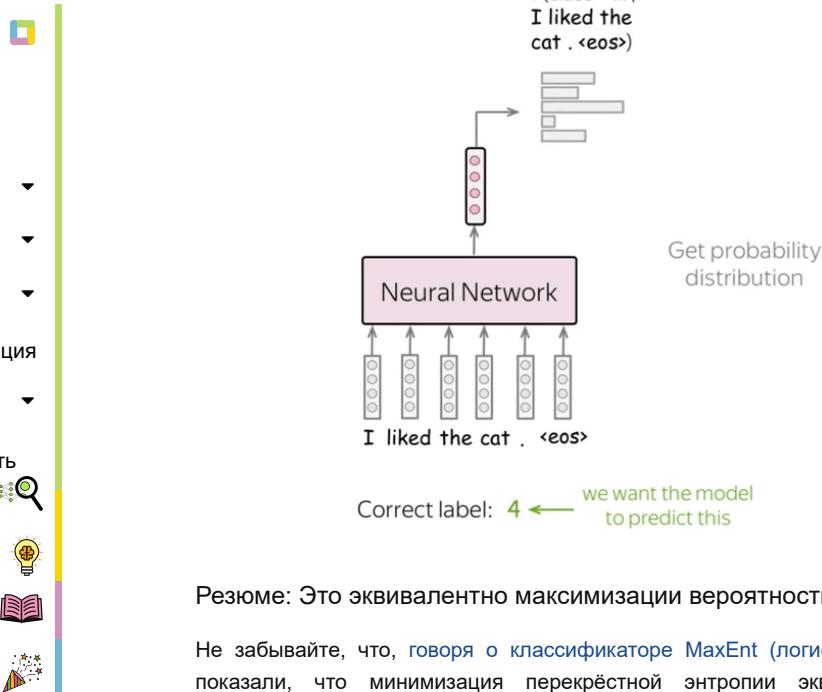
Практические советы

Анализ и интерпретируемость

Исследовательское мышление

Связанные статьи

Веселиться!



Резюме: Это эквивалентно максимизации вероятности данных.

Не забывайте, что, говоря о классификаторе MaxEnt (логистической регрессии), мы показали, что минимизация перекрёстной энтропии эквивалентна максимизации правдоподобия данных. Поэтому здесь мы также пытаемся получить оценку максимального правдоподобия (MLE) параметров модели.

Модели классификации текста

Нам нужна модель, которая может создавать вектор фиксированного размера для входных данных разной длины.

В этой части мы рассмотрим различные способы получения векторного представления входного текста с помощью нейронных сетей. Обратите внимание: хотя входные тексты могут иметь разную длину, векторное представление текста должно иметь фиксированный размер: иначе сеть не будет «работать».

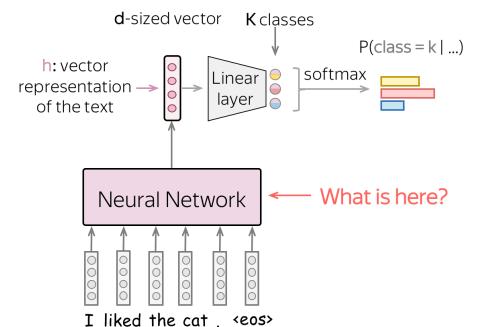
Начнём с простейших подходов, использующих только векторные представления слов (без добавления модели). Затем рассмотрим рекуррентные и свёрточные сети.

Лена: Чуть позже в ходе курса вы узнаете о Трансформерах и новейших методах классификации с использованием больших предобученных моделей.

Основы: мешок вложений (ВОЕ) и взвешенный ВОЕ

Самый простой способ — использовать только эмбеддинги слов без использования нейронной сети. Чтобы получить векторное представление текста, можно либо просуммировать все эмбеддинги токенов (Bag of Embeddings), либо использовать взвешенную сумму этих эмбеддингов (например, с весами tf-idf или другими).

Пакет векторных представлений (в идеале, вместе с наивным байесовским алгоритмом) должен быть базой для любой модели с нейронной сетью: если вы не можете добиться большего, то вообще не стоит использовать нейронные сети. Это может быть актуально, если у вас мало данных.





Классификация текста

Введение и наборы данных

Общий вид

Классические методы

Нейронные сети

Многоуровневая классификация

Практические советы

Анализ и интерпретируемость



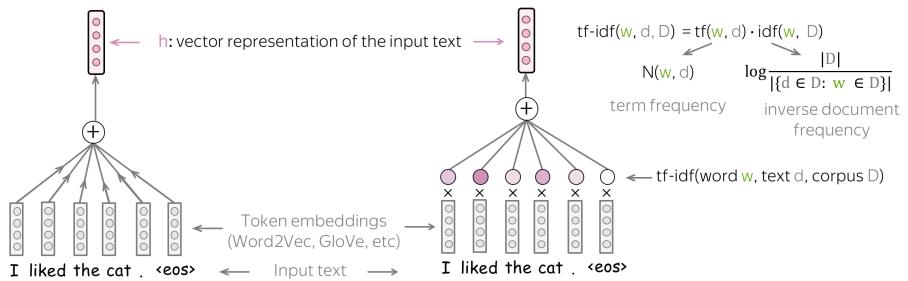
Исследовательское мышление



Связанные статьи



Веселиться!

Sum of embeddings
(Bag of Words, Bag of Embeddings)Weighted sum of embeddings
(e.g., using tf-idf weights)

Хотя мешок векторных представлений (ВОЕ) иногда называют мешком слов (BOW), обратите внимание, что эти два понятия существенно различаются. ВОЕ — это сумма векторных представлений, а BOW — сумма векторов с одним прямым значением: ВОЕ гораздо лучше разбирается в языке. Предварительно обученные векторные представления (например, Word2Vec или GloVe) распознают сходство между словами. Например, слова `awesome`, `shiny`, `great` будут представлены несвязанными признаками в BOW, но похожими векторами слов в ВОЕ.

Обратите внимание, что для использования взвешенной суммы векторных представлений необходимо придумать способ получения весов. Однако именно этого мы хотели избежать, используя нейронные сети: мы не хотим вводить ручные признаки, а позволяем сети изучать полезные закономерности.

Пакет вложений как признаки для SVM

Можно использовать SVM поверх ВОЕ! Единственное отличие от SVM в классических подходах (на основе мешка слов и мешка n-грамм) заключается в выборе ядра: здесь ядро RBF лучше.

Модели: Рекуррентные (RNN/LSTM и т. д.)

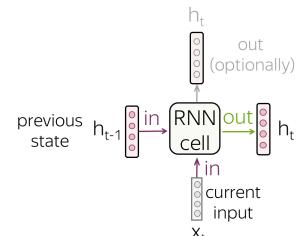
Рекуррентные сети — это естественный способ обработки текста, поскольку, подобно людям, они «считывают» последовательность токенов один за другим и обрабатывают информацию. Будем надеяться, что на каждом этапе сеть будет «запоминать» всё, что она читала ранее.

Основы: рекуррентные нейронные сети

• Ячейка RNN

На каждом шаге рекуррентная сеть получает новый входной вектор (например, встраивание токена) и предыдущее состояние сети (которое, как предполагается, кодирует всю предыдущую информацию). Используя эти входные данные, ячейка RNN вычисляет новое состояние, которое выдаёт на выходе. Это новое состояние теперь содержит информацию как о текущих входных данных, так и о данных, полученных на предыдущих шагах.

• RNN считывает последовательность токенов



Посмотрите на иллюстрацию: RNN считывает текстовый токен за токеном, на каждом шаге используя новое внедрение токена и предыдущее состояние.

Text: I like the cat on a mat <eos>
not read yet

Обратите внимание, что ячейка RNN на каждом шаге одна и та же!

• Ванильный RNN

Простейшая рекуррентная сеть, Vanilla RNN, преобразует h_{t-1} и x_t линейно, затем применяется нелинейность (чаще всего, \tanh функция):

$$h_t = \tanh(h_{t-1} W_h + x_t W_t).$$



Классификация текста

Введение и наборы данных

Общий вид

Классические методы

Нейронные сети

Многоуровневая классификация

Практические советы

Анализ и интерпретируемость



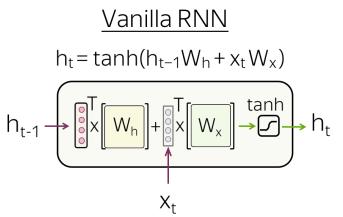
Исследовательское мышление



Связанные статьи



Веселиться!



Стандартные рекуррентные нейронные сети страдают от проблемы исчезающих и взрывных градиентов . Чтобы смягчить эту проблему, более сложные рекуррентные ячейки (например, LSTM, GRU и т.д.) выполняют несколько операций над входными данными и используют вентили. Подробнее об основах рекуррентных нейронных сетей см. в [блоге Colah](#) .

Рекуррентные нейронные сети для классификации текстов

Здесь мы (наконец-то!) рассмотрим, как использовать рекуррентные модели для классификации текстов. Всё, что вы здесь увидите, применимо ко всем рекуррентным ячейкам, а под «RNN» в этой части я подразумеваю рекуррентные ячейки в целом (например, ванильные RNN, LSTM, GRU и т. д.).

Давайте вспомним, что нам нужно:

Нам нужна модель, которая может создавать вектор фиксированного размера для входных данных разной длины.

- Просто : прочитайте текст, определите конечное состояние.

Простейшая рекуррентная модель — однослойная рекуррентная нейронная сеть (RNN). В этой сети мы выбираем состояние, которое лучше знает входной текст. Следовательно, мы должны использовать последнее состояние, поскольку только оно видело все входные токены.

- Несколько слоев : передача состояний из одной RNN в следующую

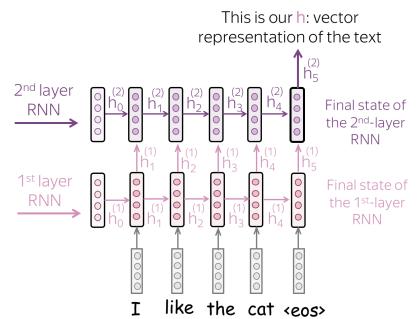
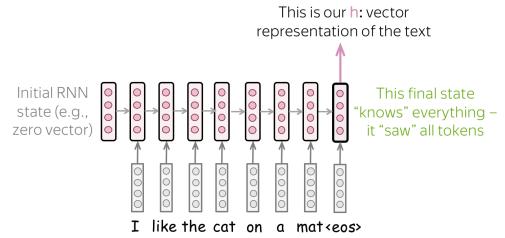
Для лучшего представления текста можно использовать несколько слоёв. В этом случае входными данными для более высокого уровня рекуррентной нейронной сети (RNN) будут данные из предыдущего слоя.

Основная гипотеза заключается в том, что при наличии нескольких слоев нижние слои будут улавливать локальные явления (например, фразы), в то время как верхние слои смогут изучать более высокоуровневые вещи (например, тему).

- Двунаправленный : использование конечных состояний из прямых и обратных RNN.

В предыдущих подходах может возникнуть проблема: последнее состояние может легко «забывать» предыдущие токены. Даже сильные модели, такие как LSTM, могут страдать от этого!

Чтобы избежать этого, мы можем использовать две рекуррентные нейронные сети: **прямую** (RNN), которая считывает входные данные слева направо, и **обратную** (RNN), которая считывает входные данные справа налево. Тогда мы сможем использовать конечные состояния обеих моделей: одна лучше запомнит конец текста, другая —



начало. Эти состояния можно объединить, суммировать или использовать как-то ещё — выбор за вами!

Классификация текста

Введение и наборы данных

Общий вид

Классические методы

Нейронные сети

Многоуровневая классификация

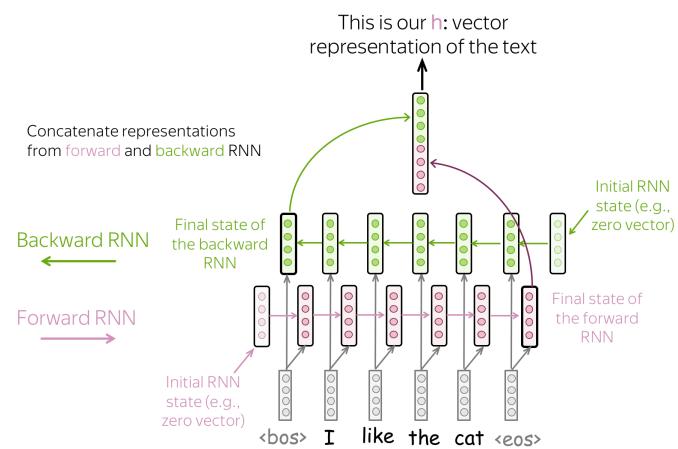
Практические советы

Анализ и интерпретируемость

Исследовательское мышление

Связанные статьи

Веселиться!



- Комбинации : делайте все, что хотите!

Вы можете комбинировать вышеперечисленные идеи. Например, в многослойной сети некоторые слои могут идти в противоположном направлении и т. д.

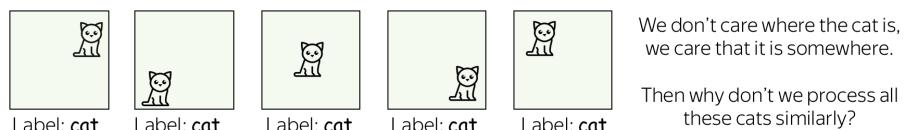
Модели: свёрточные (CNN)

Подробное описание свёрточных моделей в целом приведено в книге «Convolutional Models Supplementary». В этой части мы рассматриваем только свёртки для классификации текстов.

Свертки для изображений и инвариантность трансляции

Свёрточные сети изначально были разработаны для задач компьютерного зрения. Поэтому давайте сначала разберёмся в сути свёрточных моделей для изображений.

Представьте, что мы хотим классифицировать изображение по нескольким классам, например, кошка, собака, самолёт и т. д. В этом случае, если вы найдете кошку на изображении, вам не важно, где именно на изображении находится эта кошка: вам важно только, чтобы она где-то там была.



Свёрточные сети применяют ту же операцию к небольшим участкам изображения: так они извлекают признаки. Каждая операция ищет совпадение с шаблоном, и сеть изучает, какие из них полезны. При большом количестве слоёв изучаемые шаблоны становятся всё сложнее: от линий в начальных слоях до очень сложных узоров (например, изображение кошки или собаки целиком) в верхних. Примеры можно посмотреть в разделе «Анализ и интерпретируемость».

Это свойство называется трансляционной инвариантностью : трансляцией — потому что мы говорим о сдвигах в пространстве, инвариантностью — потому что мы хотим, чтобы это не имело значения.

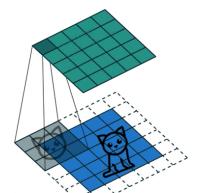


Иллюстрация адаптирована из этого замечательного репозитория .

Свертки для текста

Ну, с изображениями всё понятно: например, мы хотим иметь возможность перемещать кошку, потому что нам всё равно, где она находится. Но что насчёт текстов? На первый взгляд, всё не так просто: фразы перемещать не получится — смысл изменится, или получится что-то бессмысленное.

Однако есть и другие области применения, где можно использовать ту же интуицию. Представим, что мы хотим классифицировать тексты, но не по кошкам/собакам, как на изображениях, а по позитивной/негативной тональности. Кроме того, есть слова и фразы, которые могут быть очень информативными «подсказками» (например, « было здорово » ,

«до смерти скучно», «совершенно потрясающее», «самый лучший в жизни» и т. д.), а есть и такие, которые совершенно не важны. Нам ведь не так уж важно, где в тексте мы видим «до смерти скучно», чтобы понять его тональность, верно?

Курс НЛП | Для вас

Классификация текста

Введение и наборы данных

Общий вид

Классические методы

Нейронные сети

Многоуровневая классификация

Практические советы

Анализ и интерпретируемость

Исследовательское мышление

Связанные статьи

Веселиться!



An **absolutely great** movie! I watched the premiere with my friends.

The movie about cats was **absolutely great**, and the cats were cute.

The movie is about cats running around, and it is **absolutely great**.

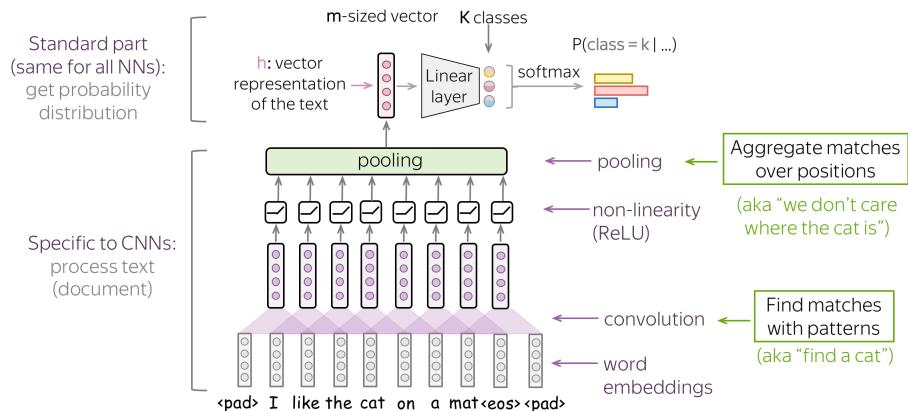
If a clue is very informative, maybe we don't care much where in a text it appears?

Типичная модель: свертка + объединение блоков

Следуя вышеизложенной интуиции, мы хотим обнаружить некоторые закономерности, но нас не особо волнует, где именно они находятся. Это поведение реализовано в два слоя:

- свертка : находит совпадения с шаблонами (как голова кошки, которую мы видели выше);
- объединение : объединяет эти совпадения по позициям (локально или глобально).

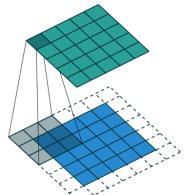
Типичная свёрточная модель для классификации текста представлена на рисунке. Для получения векторного представления входного текста к встраиванию слов применяется свёрточный слой, за которым следуют нелинейность (обычно ReLU) и операция объединения. Использование этого представления для классификации аналогично другим сетям.



Далее мы подробно обсудим основные строительные блоки, свертку и объединение, а затем рассмотрим модификации моделирования.

Основы: свёрточный слой для текста

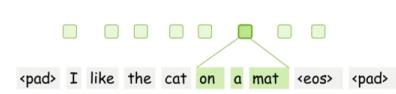
Свёрточные нейронные сети изначально разрабатывались для задач компьютерного зрения, например, для классификации изображений (кошки против собак и т. д.). Идея свёртки заключается в том, чтобы обработать изображение скользящим окном и применить к каждому окну одну и ту же операцию — фильтр свёртки.



Фильтр свёртки для изображений.
Иллюстрация взята из [этого замечательного репозитория](#).

Иллюстрация (взятая из [этого замечательного репозитория](#)) показывает этот процесс для одного фильтра: внизу — входное изображение, вверху — выход фильтра. Поскольку изображение имеет два измерения (ширина и высоту), свёртка двумерная.

В отличие от изображений, тексты имеют только одно измерение: здесь свертка одномерна: посмотрите на иллюстрацию.



Фильтр свёртки для текста.

Свертка — это линейная операция, применяемая к каждому окну.



Классификация текста

Введение и наборы данных

Общий вид

Классические методы

Нейронные сети

Многоуровневая классификация

Практические советы

Анализ и интерпретируемость



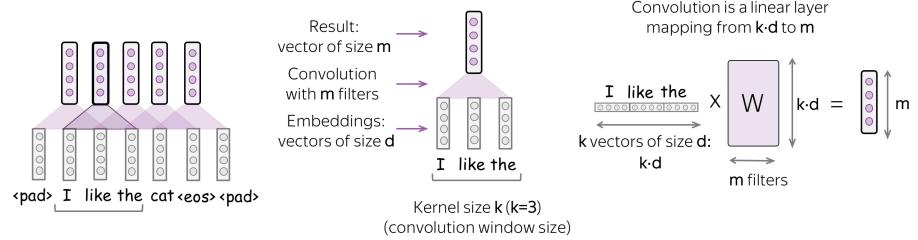
Исследовательское мышление



Связанные статьи



Веселиться!



Свёртка — это линейный слой (с последующим нелинейным слоем), применяемый к каждому входному окну. Формально предположим, что

- (x_1, \dots, x_n) - представления входных слов, $x_i \in \mathbb{R}^d$;
- d (входные каналы) - размер входного вложения;
- k (размер ядра) - длина окна свертки (на иллюстрации, $k = 3$);
- m (выходные каналы) - количество фильтров свертки (т.е. количество каналов, полученных в результате свертки).

Тогда свёртка — это линейный слой $W \in \mathbb{R}^{(k \cdot d) \times m}$. Для окно размером с окно (x_i, \dots, x_{i+k-1}) , свёртка представляет собой объединение этих векторов

$$u_i = [x_i, \dots, x_{i+k-1}] \in \mathbb{R}^{k \cdot d}$$

и умножается на матрицу свёртки:

$$F_i = u_i \times W.$$

Свёртка выполняется по входным данным со скользящим окном и применяет одно и то же линейное преобразование к каждому окну.

Интуиция: каждый фильтр извлекает признак

Интуитивно понятно, что каждый фильтр в свёртке извлекает признак.

- Один фильтр — один экстрактор признаков

Фильтр принимает векторные представления в текущем окне и линейно преобразует их в единый признак. Формально, для окна $u_i = [x_i, \dots, x_{i+k-1}] \in \mathbb{R}^{k \cdot d}$ фильтр $f \in \mathbb{R}^{k \cdot d}$ вычисляет скалярное произведение:

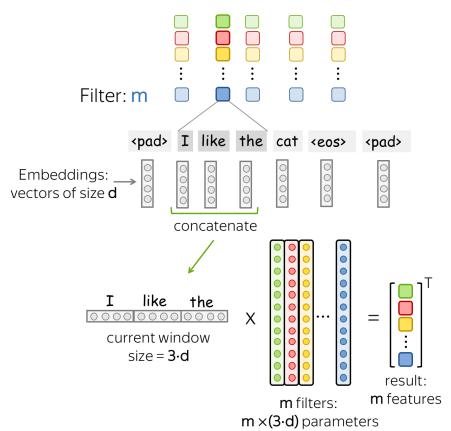
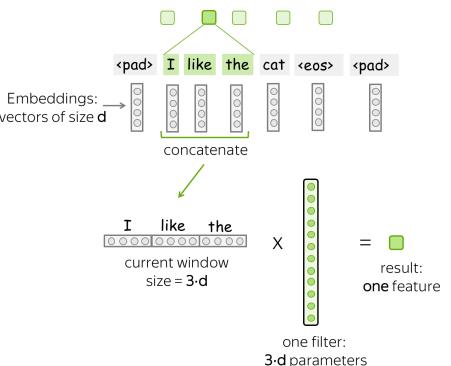
$$F_i^{(f)} = (f, u_i).$$

Число $F_i^{(f)}$ (извлеченный «признак») является результатом применения фильтра f окну (x_i, \dots, x_{i+k-1}) .

- m фильтров : m экстракторов признаков



Один фильтр извлекает один признак. Обычно нам требуется много признаков: для этого приходится использовать несколько фильтров. Каждый фильтр считывает входной текст и извлекает свой признак — см. иллюстрацию. Количество фильтров — это количество выходных признаков, которые вы хотите получить. Число фильтров вместо одного, размер свёрточного слоя, который мы обсуждали выше, станет $(k \cdot d) \times m$.





Классификация текста

Введение и наборы данных

Общий вид

Классические методы

Нейронные сети

Многоуровневая классификация

Практические советы

Анализ и интерпретируемость



Исследовательское мышление



Связанные статьи



Веселиться!



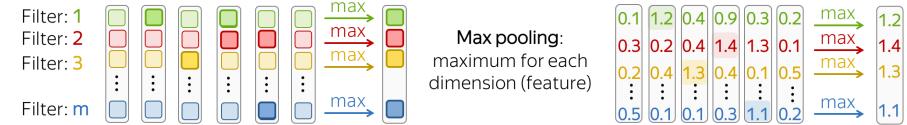
Это делается параллельно! Обратите внимание: хотя я показываю, как свёрточная нейронная сеть «читывает» текст, на практике эти вычисления выполняются параллельно.

Основы: операция объединения

После извлечения свертки для каждого окна, объединяющий слой суммирует признаки в определённой области. Объединение слоев используется для уменьшения размерности входных данных и, следовательно, для уменьшения количества параметров, используемых сетью.

- Объединение максимальных и средних значений

Наиболее популярным является метод максимального пула: он берет максимум по каждому измерению, т.е. берет максимальное значение каждой характеристики.

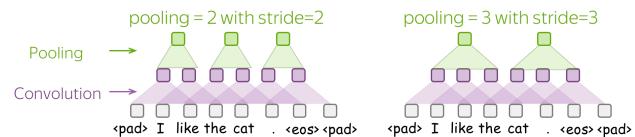


Интуитивно понятно, что каждый признак «срабатывает», когда обнаруживает некий шаблон: визуальный шаблон на изображении (линию, текстуру, кошачью лапку и т. д.) или текстовый шаблон (например, фразу). После операции объединения мы получаем вектор, показывающий, какой из этих шаблонов встречался во входных данных.

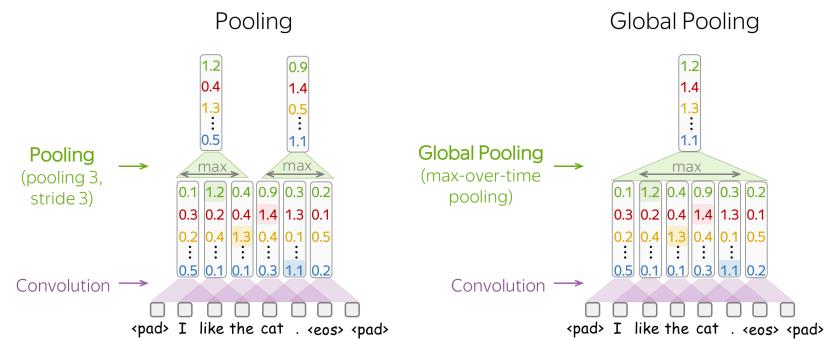
Объединение средних значений работает аналогичным образом, но для каждого признака вычисляется среднее значение, а не максимум.

- Объединение и глобальное объединение

Подобно свёртке, пулинг применяется к окнам, состоящим из нескольких элементов. Пулинг также имеет параметр шага, и наиболее распространённый подход — использовать пулинг с неперекрывающимися окнами. Для этого необходимо установить параметр шага равным размеру пула. См. иллюстрацию.



Разница между пулингом и глобальным пулингом заключается в том, что пулинг применяется к признакам в каждом окне независимо, в то время как глобальный пулинг выполняется для всего входного набора. Для текстов глобальный пулинг часто используется для получения одного вектора, представляющего весь текст; такой глобальный пулинг называется пулингом с максимальным временем, где ось времени идёт от первого входного токена к последнему.



Свёрточные нейронные сети для классификации текстов

Теперь, когда мы разобрались, как работают свёртка и пулинг, перейдём к модификациям моделирования. Для начала вспомним, что нам понадобится:

Нам нужна модель, которая может создавать вектор фиксированного размера для входных данных разной длины.

Классификация текста

Введение и наборы данных

Общий вид

Классические методы

Нейронные сети

Многоуровневая классификация

Практические советы

Анализ и интерпретируемость

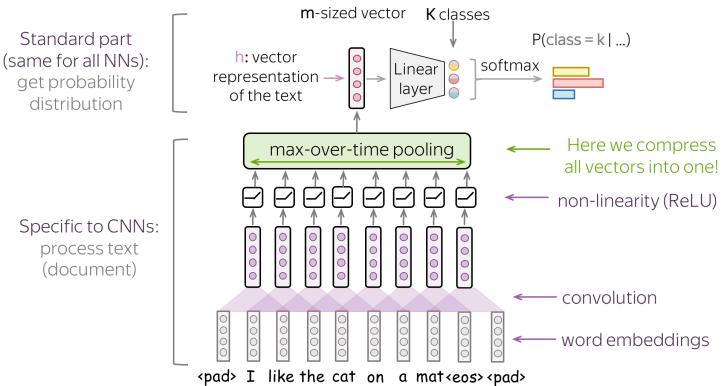
Исследовательское мышление

Связанные статьи

Веселиться!

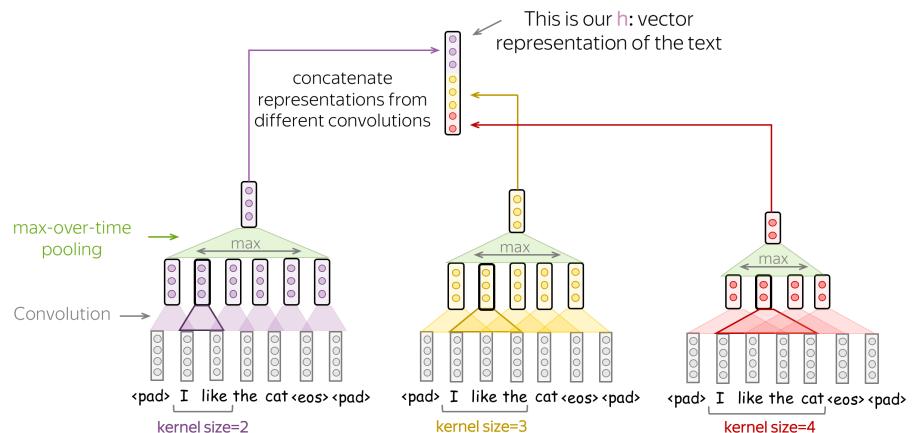
Поэтому нам необходимо построить свёрточную модель, представляющую текст как один вектор.

Базовая свёрточная модель для классификации текста показана на рисунке. Она практически такая же, как и та, что мы видели ранее: единственное изменение — это указание типа используемого пулинга. В частности, после свёртки мы используем глобальный пулинг по времени. Это ключевая операция: она позволяет сжать текст в один вектор. Сама модель может быть разной, но в определённый момент ей приходится использовать глобальный пулинг для сжатия входных данных в один вектор.



- Несколько сверток с разными размерами ядра

Вместо того, чтобы выбирать один размер ядра для свёртки, вы можете использовать несколько свёрток с разными размерами ядра. Рецепт прост: применить каждую свёртку к данным, добавить нелинейность и глобальное объединение после каждой из них, а затем объединить результаты (на иллюстрации нелинейность опущена для простоты). Так вы получите векторное представление данных, используемое для классификации.

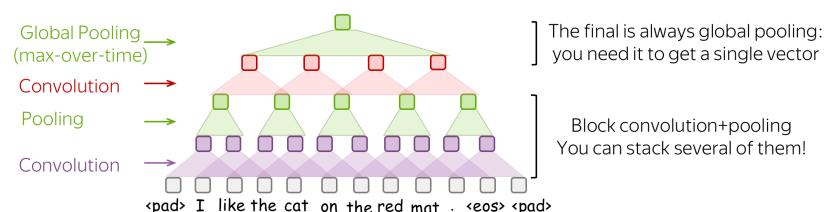


Эта идея была использована, среди прочего, в статье « [Свёрточные нейронные сети для классификации предложений](#) » и многих последующих работах.

- Стек из нескольких блоков: свертка + объединение

Вместо одного слоя можно наложить друг на друга несколько блоков свёртки и пулинга. После нескольких блоков можно применить ещё одну свёртку, но уже с глобальным пулингом. Помните: вам нужен один вектор фиксированного размера — для этого вам понадобится глобальный пулинг.

Такие многослойные свёртки могут быть полезны, когда тексты очень длинные, например, если ваша модель является уровнем символов (а не на уровне слов).



Эта идея была использована, среди прочего, в статье « [Свёрточные сети на уровне символов для классификации текстов](#) » .

Многоуровневая классификация

Курс НЛП | Для вас

Классификация текста

Введение и наборы данных

Общий вид

Классические методы

Нейронные сети

Многоуровневая классификация

Практические советы

Анализ и интерпретируемость

Исследовательское мышление

Связанные статьи

Веселиться!



Классификация по нескольким меткам отличается от проблем с одной меткой, которые мы обсуждали ранее, тем, что у каждого входного параметра может быть несколько правильных меток. Например, у твита может быть несколько хэштегов, у пользователя может быть несколько тем для интереса и т. д.



Для решения проблемы с несколькими метками нам необходимо изменить две вещи в конвейере с одной меткой, который мы обсуждали ранее:

- 1.
- 2.

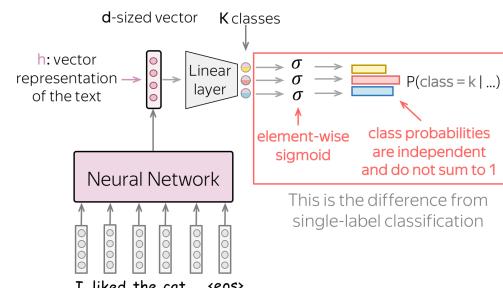
- модель (как мы оцениваем вероятности классов);
- функция потерь .

Модель: Softmax → Поэлементная сигмоида

После последнего линейного слоя имеем K значения, соответствующие K классы — это значения, которые мы должны преобразовать в вероятности классов.

Для задач с одной меткой мы использовали softmax: он преобразует K значения в распределение вероятностей, т.е. сумма всех вероятностей равна 1. Это означает, что классы имеют одинаковую массу вероятности: если вероятность одного класса высока, то другие классы не могут иметь большую вероятность (Лена : Еще раз представьте себе кучу котят, которые едят из одной миски: один котенок всегда ест за счет других).

Для задач с несколькими метками мы преобразуем каждую из K значения в вероятность соответствующего класса независимо от других. В частности, мы применяем сигмоидальную функцию $\sigma(x) = \frac{1}{1+e^{-x}}$ к каждому из K ценности.



Интуитивно мы можем думать об этом как о K независимые бинарные классификаторы, использующие одно и то же текстовое представление.

Функция потерь: двоичная кросс-энтропия для каждого класса

Функция потерь изменяется, чтобы обеспечить возможность использования нескольких меток: для каждого класса мы используем двоичную кросс-энтропийную потерю. Посмотрите на иллюстрацию.

Training example: I liked the cat on the mat <eos>

Labels: k, t

↑
target

Model prediction:

$P(y_i=1 | \dots, \text{<eos>}), i = 1..K$

Target:

$p_i^*, i = 1..K$



Binary cross-entropy loss for each class:

$$-\sum_{i=1}^K [p_i^* \cdot \log P(y_i = 1 | x) + (1 - p_i^*) \cdot \log P(y_i = 0 | x)] \rightarrow \min,$$

where $P(y_i = 0 | x) = 1 - P(y_i = 1 | x)$

binary classifier loss for class i

Практические советы

Встраивание слов: как с ними бороться?



Классификация текста

Введение и наборы данных

Общий вид

Классические методы

Нейронные сети

Многоуровневая классификация

Практические советы

Анализ и интерпретируемость



Исследовательское мышление



Связанные статьи

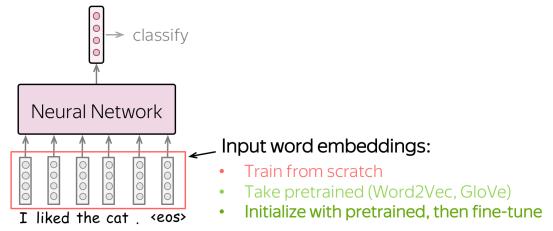


Веселиться!



Входные данные для сети представлены вложениями слов. Получить эти вложения для вашей модели можно тремя способами:

- обучение с нуля как часть вашей модели,
- взять предобученные (Word2Vec, GloVe и т.д.) и исправить их (использовать как статические векторы),
- инициализировать с помощью предварительно обученных вложений и обучите их с помощью сети («тонкая настройка»).



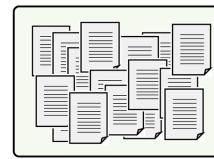
- Train from scratch
- Take pretrained (Word2Vec, GloVe)
- Initialize with pretrained, then fine-tune

Давайте рассмотрим эти варианты, взглянув на данные, которые может использовать модель. Обучающие данные для классификации размечены и специфичны для конкретной задачи, но размеченные данные обычно сложно получить. Поэтому этот корпус, вероятно, будет не очень большим (как минимум), или не будет разнообразным, или и тем, и другим. Напротив, обучающие данные для векторных представлений слов не размечены — достаточно простых текстов. Следовательно, эти наборы данных могут быть огромными и разнообразными, что даёт массу возможностей для изучения.



Training data for text classification (labeled)

- Not huge, or not diverse, or both
- Domain: task-specific



Training data for word embeddings (unlabeled)

- Huge diverse corpus (e.g., Wikipedia)
- Domain: general

Теперь давайте подумаем, что будет знать модель в зависимости от того, что мы делаем с эмбеддингами. Если эмбеддинги обучаются с нуля, модель будет «знать» только данные классификации — этого может быть недостаточно для эффективного изучения связей между словами. Но если мы используем предобученные эмбеддинги, они (и, следовательно, вся модель) будут знать огромный корпус — они узнают много нового о мире. Чтобы адаптировать эти эмбеддинги к данным, специфичным для вашей задачи, вы можете точно настроить их, обучая всю сеть — это может дать прирост производительности (хотя и незначительный).

- Train from scratch

What they will know:



May be not enough to learn relationships between words

- Take pretrained (Word2Vec, GloVe)

What they will know:



Know relationships between words, but are not specific to the task

- Initialize with pretrained, then fine-tune

What they will know:



Know relationships between words and adapted for the task

"Transfer" knowledge from a huge unlabeled corpus to your task-specific model

Использование предобученных вложений является примером переноса обучения: посредством вложений мы «переносим» знания из их обучающих данных в нашу модель, специфичную для конкретной задачи. Мы подробнее рассмотрим перенос обучения далее в курсе.



Тонкая настройка предобученных векторных моделей или нет? Прежде чем приступить к обучению моделей, подумайте, почему тонкая настройка может быть полезна и для каких типов примеров она может быть полезна. Подробнее об этом упражнении можно узнать в разделе «Исследовательское мышление».



Более подробную информацию и эксперименты с различными настройками для встраивания слов можно найти [в этом резюме статьи](#).

Дополнение данных: получите больше данных бесплатно



Классификация текста

Введение и наборы данных

Общий вид

Классические методы

Нейронные сети

Многоуровневая классификация

Практические советы

Анализ и интерпретируемость



Исследовательское мышление



Связанные статьи



Веселиться!



Аугментация данных изменяет ваш набор данных различными способами, чтобы получить альтернативные версии одного и того же обучающего примера. Аугментация данных может повысить

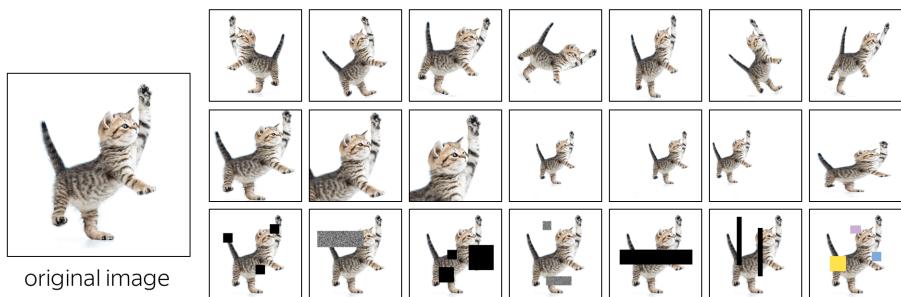
- Объем данных.

Качество вашей модели во многом зависит от ваших данных. Для моделей глубокого обучения наличие больших наборов данных очень (очень!) важно.

- Разнообразие данных.

Представляя различные версии обучающих примеров, вы обучаете модель быть более устойчивой к реальным данным, которые могут быть более низкого качества или просто немного отличаться от ваших обучающих данных. С дополненными данными модель менее склонна к переобучению на определённых типах обучающих примеров и будет больше полагаться на общие закономерности.

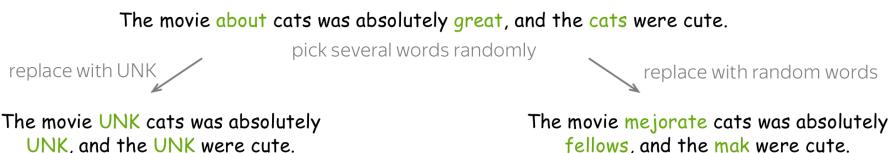
Дополнение данных изображений выполняется легко: взгляните на примеры ниже. Стандартные дополнения включают в себя переворачивание изображения, геометрические преобразования (например, поворот и растяжение в определённом направлении), а также наложение различных заплаток на части изображения.



Как можно сделать что-то подобное для текстов?

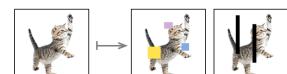
- пропуск слов — самый простой и популярный

Исключение слов — это простейшая регуляризация: для каждого примера вы выбираете несколько слов случайным образом (скажем, каждое слово выбирается с вероятностью 10%) и заменяете выбранные слова либо специальным токеном UNK , либо случайным токеном из словаря.



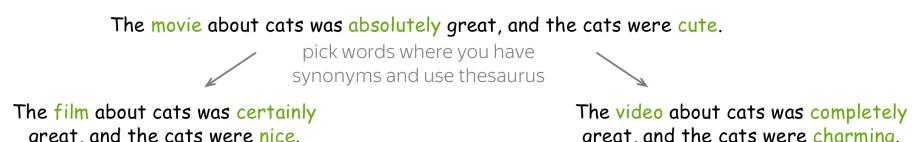
Мотивация здесь проста: мы учим модель не полагаться чрезмерно на отдельные токены, а учитывать контекст всего текста. Например, здесь мы замаскировали слово «great» , и модель должна понимать его эмоциональную окраску, основываясь на других словах.

Примечание: для изображений это соответствует маскированию некоторых областей. Маскируя область изображения, мы также хотим, чтобы модель не слишком полагалась на локальные особенности и использовала более глобальный контекст.



- использовать внешние ресурсы (например, тезаурус) — немного сложнее

Немного более сложный подход — замена слов или фраз их синонимами. Сложность заключается в получении этих синонимов: для этого нужны внешние ресурсы, которые редко доступны для языков, отличных от английского (для английского можно использовать, например, WordNet). Другая проблема заключается в том, что для языков с богатой морфологией (например, русского) вы, скорее всего, нарушите грамматическое соответствие.





Классификация текста

Введение и наборы данных

Общий вид

Классические методы

Нейронные сети

Многоуровневая классификация

Практические советы

Анализ и интерпретируемость



Исследовательское мышление



Связанные статьи



Веселиться!



- использовать отдельные модели — еще сложнее

Ещё более сложный метод — перефразировать целые предложения с помощью внешних моделей. Популярный метод перефразирования — это перевод предложения на другой язык и обратно. Мы научимся обучать модель перевода чуть позже (в лекции «Seq2seq и внимание»), но пока можно использовать промышленные системы, например, Яндекс.Переводчик, Google Переводчик и т. д. (Лена: Конечно, лично я склоняюсь к Яндексу :)). Обратите внимание, что можно комбинировать системы перевода и языки, чтобы получить несколько парадигм.

The movie about cats was absolutely great, and the cats were cute.

En-Ru ↓ Yandex Translate

Фильм о кошках был замечательный, и кошки были милые.

Ru-En ↓ Yandex Translate

The cat movie was just great, and the cats were cute.

En-Zh ↓ Google Translate

關於貓的電影絕對很棒，而且貓很可愛。

Zh-En ↓ Google Translate

Movies about cats are absolutely great, and cats are cute.

En-Ja ↓ Google Translate

猫の映画は本当に素晴らしい、猫はかわいい。

Ja-En ↓ Yandex Translate

The Cat movie is really nice and the cat is cute.



Примечание: для изображений последние два метода соответствуют геометрическим преобразованиям: мы хотим изменить текст, сохранив его смысл. Это отличается от исключения слов, при котором некоторые части текста полностью теряются.

🔍 Анализ и интерпретируемость

Чему учатся свертки? Анализ сверточных фильтров

Свёртки в компьютерном зрении: визуальные паттерны

Свёртки изначально разрабатывались для изображений, и уже довольно хорошо понятно, что именно улавливают фильтры и как работают фильтры разных слоёв иерархии. В то время как нижние слои захватывают простые визуальные узоры, такие как линии или круги, верхние слои могут захватывать целые изображения, животных, людей и т. д.



Примеры паттернов, полученных с помощью фильтров свёртки изображений. Примеры взяты из [Activation Atlas](#) с сайта [distill.pub](#).

А как насчет сверток в текстах?

Эта часть основана на статье «[Понимание сверточных нейронных сетей для классификации текстов](#)».

Фильтры улавливают локальные визуальные закономерности изображений, важные для классификации. Для текста такими локальными закономерностями являются n-граммы слов. Основные выводы о работе сверточных нейронных сетей с текстами:

- Свёрточные фильтры используются в качестве детекторов n-грамм
Каждый фильтр специализируется на одном или нескольких семействах тесно связанных n-грамм. Фильтры неоднородны, то есть один фильтр может, и часто действительно обнаруживает, несколько совершенно разных семейств n-грамм.
- Максимальное объединение данных (max-pooling) индуцирует пороговое поведение.
Значения ниже заданного порога игнорируются при построении прогноза (т.е. не имеют отношения к нему). Например, в [этой статье](#) показано, что в среднем 40% объединенных n-грамм можно отбросить без потери производительности.

Самый простой способ понять, что улавливает сеть, — это посмотреть, какие паттерны активируют её нейроны. Для свёрток мы выбираем фильтр и находим те n-граммы,

Классификация текста

Введение и наборы данных

Общий вид

Классические методы

Нейронные сети

Многоуровневая классификация

Практические советы

Анализ и интерпретируемость

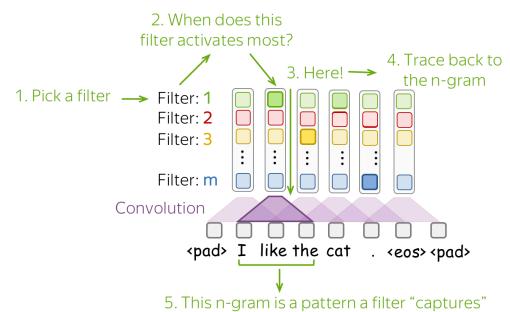
Исследовательское мышление

Связанные статьи

Веселиться!

которые активируют этот фильтр чаще всего.

Ниже приведены примеры n-грамм top-1 для нескольких фильтров. Для одного из них мы также показываем другие n-граммы, приводящие к высокой активации этого фильтра — можно заметить, что n-граммы имеют очень схожее значение.



filter	Top n-gram	Score
1	poorly designed junk	7.31
2	simply would not	5.75
3	a minor drawback	6.11
4	still working perfect	6.42
5	absolutely gorgeous .	5.36
6	one little hitch	5.72
7	utterly useless .	6.33
8	deserves four stars	5.56
9	a mediocre product	6.91

Top n-grams for filter 4		Score
1	still working perfect	6.42
2	works - perfect	5.78
3	isolation proves invaluable	5.61
4	still near perfect	5.6
5	still working great	5.45
6	works as good	5.44
7	still holding strong	5.37

A filter activates for a family of n-grams with similar meaning

Более подробную информацию можно найти в статье « [Понимание сверточных нейронных сетей для классификации текстов](#) ».

Как насчет классификаторов RNN?



Как RNN обучаются классификации текстовых процессов? Узнайте [здесь](#).



Исследовательское мышление

Как

- Прочтите краткое описание в начале — это наша отправная точка, нечто известное.
- Прочтите вопрос и подумайте: минуту, день, неделю... — дайте себе время! Даже если вы не думаете об этом постоянно, что-то всё равно может прийти в голову.
- Посмотрите на возможные ответы — предыдущие попытки решить эту задачу. Важно не обязательно предлагать что-то в точности похожее — помните, каждая статья обычно занимает у авторов несколько месяцев. Важна привычка думать о таких вещах! Всё остальное, что нужно учёному, — это время: пробовать, ошибаться, думать, пока не получится.

Известно, что изучение чего-либо проходит легче, если не сразу получить ответ, а сначала подумать над ним. Даже если вы не хотите становиться исследователем, это всё равно хороший способ учиться!

Классические подходы

Улучшить наивный байесовский алгоритм

Простейшая реализация наивного байесовского алгоритма использует токены в качестве признаков. Однако это не всегда хорошо: совершенно разные тексты могут иметь одинаковые признаки.

This is **rather good**: not bad at all! This is **rather bad**: not good at all!

this, is, good, bad, rather, not, at, all!

Naïve Bayes features

? В примере выше мы видим главную проблему наивного байесовского алгоритма: он ничего не знает о контексте. Конечно, мы не можем избавиться от



Классификация текста

Введение и наборы данных

Общий вид

Классические методы

Нейронные сети

Многоуровневая классификация

Практические советы

Анализ и интерпретируемость



Исследовательское мышление



Связанные статьи



Веселиться!



«наивных» предположений (иначе это уже не будет наивным байесовским алгоритмом). Но можем ли мы улучшить процесс извлечения признаков?

► Возможные ответы

? Какие еще типы функций вы можете придумать?

► Возможные ответы

? Все ли слова одинаково необходимы для классификации? Если нет, как можно модифицировать метод?

► Возможные ответы

○ Нейронные подходы

Тонкая настройка вложений: почему и когда это может помочь?

Embeddings of antonyms are often very close.

Прежде чем обучать модели, подумайте, почему тонкая настройка может быть полезна и какие типы примеров могут быть ей полезны. Вспомните, как обучаются вложения: слова, которые используются в текстах схожим образом, имеют очень близкие вложения. Поэтому иногда антонимы оказываются наиболее близкими друг к другу, например, *descent* и *ascent*.

Any problems for sentiment analysis?



? Представьте, что мы хотим использовать вложения для классификации тональности. Можете ли вы найти примеры антонимов, если их вложения очень близки, это повредит классификации тональности? Если да, значит, это может быть полезно для более точной настройки!

► Возможные ответы

Здесь будет больше упражнений!



Эта часть будет время от времени расширяться.



Связанные статьи

Как

- Высокий уровень : просмотрите основные результаты в кратких обзورах – получите представление о том, что происходит в данной области.
- Немного глубже : по темам, которые вас интересуют, читайте более подробные обзоры с иллюстрациями и пояснениями. Просмотрите ход рассуждений авторов и их ключевые наблюдения.
- Подробный анализ : прочитайте понравившиеся статьи. Теперь, когда вы поняли основную идею, будет проще!



Классификация текста

Введение и наборы данных

Общий вид

Классические методы

Нейронные сети

Многоуровневая классификация

Практические советы

Анализ и интерпретируемость



Исследовательское мышление



Связанные статьи



Веселиться!



Что внутри:

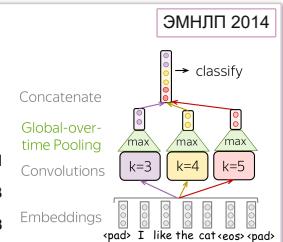
- Свертки для классификации: Классика
- Анализ рекуррентных нейронных сетей для классификации тональности
- ...будет обновлено

○ Свертки для классификации: Классика

Свёрточные нейронные сети для классификации предложений

Юн Ким

Даже очень простая сверточная нейронная сеть с одним слоем поверх векторных представлений слов демонстрирует очень хорошую производительность (без признаков, требующих каких-либо внешних знаний!). В статье также показана важность использования предобученных векторных представлений (а не обучения с нуля) и преимущества тонкой настройки.

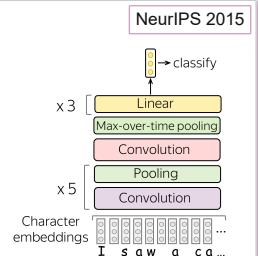


▶ Подробнее

Character-level Convolutional Networks for Text Classification

Xiang Zhang, Junbo Zhao, Yann LeCun

This is the first paper showing that CNNs only on characters can do quite well. This is interesting: classification can be done without any external knowledge, even without text segmentation into words! An important point is that character-level CNNs can do better than classical approaches only for large datasets.



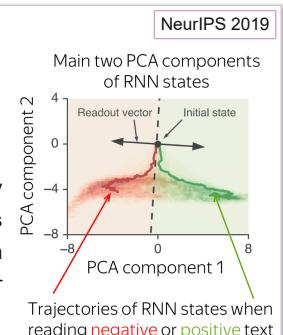
▶ More details

○ Analyzing RNNs for Sentiment Classification

Reverse engineering recurrent networks for sentiment classification reveals line attractor dynamics

Niru Maheswaranathan, Alex H. Williams, Matthew D. Golub, Surya Ganguli, David Sussillo

If we take an RNN trained for sentiment analysis and apply PCA to lots of its states, we'll see that almost all variance is explained with only two components. Moreover, when such RNN reads a text, its states move along a 1D plane in either negative or positive direction depending on the word it reads.



▶ More details



Классификация текста

Введение и наборы данных

Общий вид

Классические методы

Нейронные сети

Многоуровневая классификация

Практические советы

Анализ и интерпретируемость



Исследовательское
мышление



Связанные статьи



Веселиться!



Здесь будет еще больше
статьй!



Статьи будут появляться постепенно.

Веселиться!

Вскоре!



Мы все еще работаем над этим!

Последнее обновление: 12 июня 2025 г.



Последовательность последовательностью (seq2seq) и внимание

Самая популярная задача перевода текста из одной последовательности в другую — это перевод: обычно с одного естественного языка на другой. За

En
I saw a cute cat.Zh
我看到一只可爱的猫。

последние пару лет коммерческие системы стали на удивление хорошо справляться с машинным переводом — например, [Google Translate](#), [Yandex Translate](#), [DeepL Translator](#), [Bing](#) и [Microsoft Translator](#). Сегодня мы рассмотрим основные функции этих систем.

Помимо популярного машинного перевода между естественными языками, вы можете переводить между языками программирования (см., например, публикацию в блоге Facebook AI « [Глубокое обучение переводу между языками программирования](#) ») или между любыми последовательностями токенов, которые вы можете придумать. В дальнейшем под машинным переводом мы будем понимать любую общую задачу перевода из одной последовательности в другую, то есть перевод между последовательностями токенов любой природы.

Далее мы сначала познакомимся с основами seq2seq, затем поговорим о внимании — неотъемлемой части всех современных систем, и, наконец, рассмотрим самую популярную модель — Transformer. Конечно же, с большим количеством анализа, упражнений, статей и увлекательного времяпрепровождения!

Основы последовательности

Формально, в задаче машинного перевода мы имеем входную последовательность x_1, x_2, \dots, x_m и выходную последовательность y_1, y_2, \dots, y_n (обратите внимание, что их длины могут быть разными). Трансляцию можно рассматривать как поиск целевой последовательности, наиболее вероятной при заданных входных данных; формально это целевая последовательность, которая максимизирует условную вероятность $p(y|x)$:

$$y^* = \arg \max_y p(y|x).$$

Если вы двуязычны и можете легко переводить между языками, у вас есть интуитивное чувство $p(y|x)$ и может сказать что-то вроде : «...ну, этот перевод более естественен для этого предложения». Но при машинном переводе мы усваиваем функцию $p(y|x, \theta)$ с некоторыми параметрами θ , а затем найти его argmax для заданного ввода:

$$y' = \arg \max_y p(y|x, \theta).$$

Human Translation

$$y^* = \arg \max_y p(y|x)$$

The “probability” is intuitive and is given by a human translator’s expertise

Machine Translation

$$y' = \arg \max_y p(y|x, \theta)$$

Questions we need to answer

- modeling

How does the model for $p(y|x, \theta)$ look like?

- learning

How to find θ ?

- search

How to find the argmax?

Чтобы определить систему машинного перевода, нам необходимо ответить на три вопроса:

- моделирование - как модель для $p(y|x, \theta)$ выглядит как?
- обучение - как найти параметры θ ?
- вывод - как найти лучшее y ?

В этом разделе мы дадим исчерпывающие ответы на второй и третий вопросы, но рассмотрим лишь простейшую модель. Более «реальные» модели будут рассмотрены далее в разделах «[Внимание](#)» и «[Трансформатор](#)».

Структура кодировщика-декодера

Курс НЛП | Для вас



Seq2seq и внимание

Основы Seq2seq

Внимание

Трансформатор

Сегментация подслов: BPE

Анализ и интерпретируемость

Исследовательское мышление

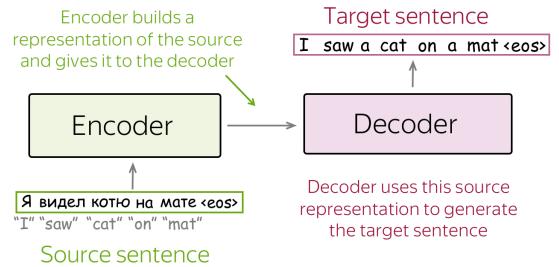
Связанные статьи

Веселиться!

Кодер-декодер — стандартная парадигма моделирования для задач последовательного преобразования. Эта структура состоит из двух компонентов:

- кодер - считывает исходную последовательность и создает ее представление;
- декодер - использует исходное представление от кодера для генерации целевой последовательности.

В этой лекции мы рассмотрим разные модели, но все они имеют структуру кодера-декодера.



Модели условного языка

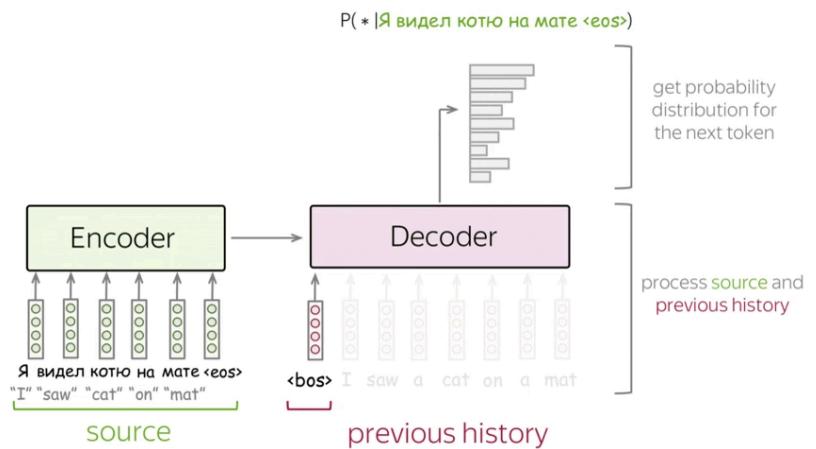
На лекции [по языковому моделированию](#) мы научились оценивать вероятность $p(y)$ последовательностей токенов $y = (y_1, y_2, \dots, y_n)$. В то время как языковые модели оценивают безусловную вероятность $p(y)$ последовательности y , модели последовательностей должны оценивать условную вероятность $p(y|x)$ последовательности y учитывая источник x . Именно поэтому задачи «последовательность-последовательность» можно моделировать с помощью условных языковых моделей (CLM) — они работают аналогично LM, но дополнительно получают исходную информацию x .

$$\text{Language Models: } P(y_1, y_2, \dots, y_n) = \prod_{t=1}^n p(y_t | y_{<t})$$

$$\text{Conditional Language Models: } P(y_1, y_2, \dots, y_n | x) = \prod_{t=1}^n p(y_t | y_{<t}, x)$$

condition on source x

Лена : Обратите внимание, что моделирование условного языка — это нечто большее, чем просто способ решения задач «последовательность-последовательность». В самом общем смысле, x может быть чем-то иным, чем последовательность токенов. Например, в задании «Подпись к изображению» x это изображение и y — описание этого изображения.



Поскольку единственное отличие от LM — это наличие источника x Моделирование и обучение очень похожи на языковые модели. В частности, высокоуровневый конвейер выглядит следующим образом:

- передавать исходные и ранее сгенерированные целевые слова в сеть;
- получить векторное представление контекста (как исходного, так и предыдущего целевого) из декодера сетей;
- на основе этого векторного представления предсказать распределение вероятностей для следующего токена.



Seq2seq и внимание

Основы Seq2seq

Внимание

Трансформатор

Сегментация подслов: BPE

Анализ и интерпретируемость



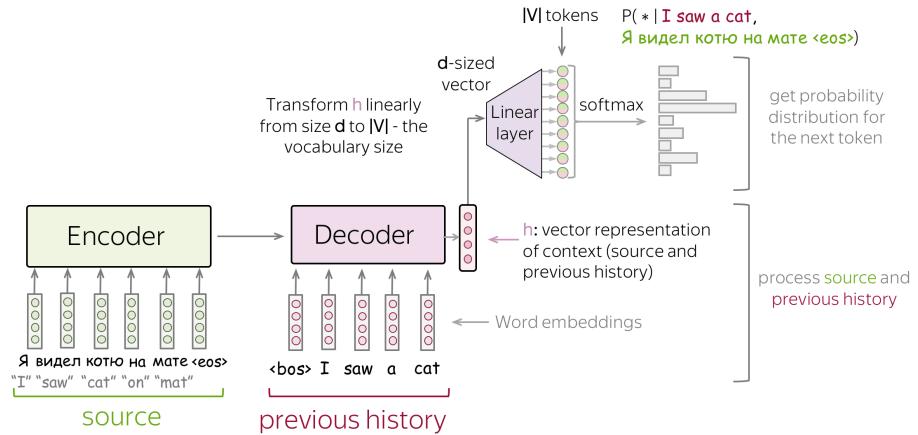
Исследовательское мышление



Связанные статьи

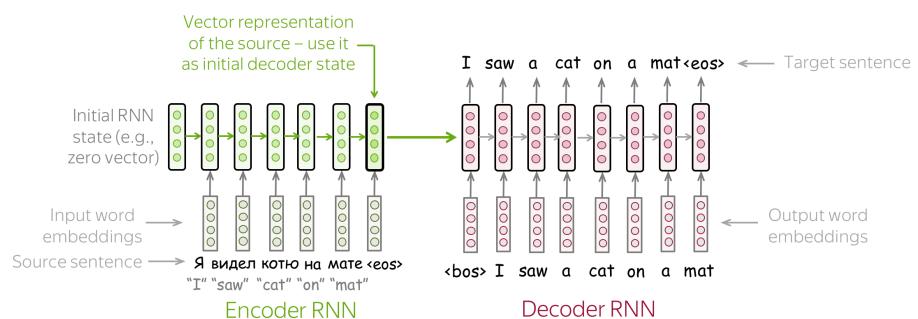


Веселиться!



Подобно нейронным классификаторам и языковым моделям, мы можем представить себе классификационную часть (то есть, как получить вероятности токенов из векторного представления текста) очень просто. Векторное представление текста имеет некоторую размерность d , но в конечном итоге нам нужен вектор размером $|V|$ (вероятности для $|V|$ токены/классы). Чтобы получить $|V|$ -размерный вектор из d , мы можем использовать линейный слой. Как только у нас есть $|V|$ -вектор размером с вектор, все, что осталось, это применить операцию softmax для преобразования необработанных чисел в вероятности токенов.

Простейшая модель: две рекуррентные нейронные сети для кодера и декодера



Простейшая модель кодер-декодер состоит из двух рекуррентных нейронных сетей (LSTM): одной для кодера и другой для декодера. Кодер-RNN считывает исходное предложение, а его конечное состояние используется в качестве начального состояния декодера. Предполагается, что конечное состояние кодера «кодирует» всю информацию об источнике, и декодер сможет сгенерировать целевое предложение на основе этого вектора.

Эта модель может иметь различные модификации: например, кодер и декодер могут иметь несколько слоёв. Такая модель с несколькими слоями использовалась, например, в статье « [Sequence to Sequence Learning with Neural Networks](#) » — одной из первых попыток решения задач преобразования последовательностей в последовательности с помощью нейронных сетей.

В той же статье авторы рассмотрели последнее состояние кодировщика и визуализировали несколько примеров (см. ниже). Интересно, что представления предложений со схожим значением, но разной структурой очень похожи!



Примеры взяты из статьи « [Последовательное обучение с помощью нейронных сетей](#) » .



В статье «[Последовательное обучение с помощью нейронных сетей](#)» был представлен элегантный приём, позволяющий улучшить работу такой простой модели LSTM. Подробнее об этом читайте в этом [упражнении](#) в разделе «[Исследовательское мышление](#)».

Seq2seq и внимание

Основы Seq2seq

Внимание

Трансформатор

Сегментация подслов: BPE

Анализ и интерпретируемость

Исследовательское мышление

Связанные статьи

Веселиться!

Обучение: потеря кросс-энтропии (еще раз)

Лена: Это та же самая потеря кросс-энтропии, которую мы обсуждали ранее на лекциях по [классификации текстов](#) и [моделированию языка](#) — вы можете пропустить эту часть или пройти ее довольно легко :)

Подобно нейронным LM, нейронные модели seq2seq обучаются предсказывать распределение вероятностей следующего токена с учётом предыдущего контекста (исходного и предыдущего целевого токенов). Интуитивно понятно, что на каждом шаге мы максимизируем вероятность, которую модель присваивает правильному токену.

Формально предположим, что у нас есть обучающий экземпляр с источником $x = (x_1, \dots, x_m)$ и целью $y = (y_1, \dots, y_n)$. Тогда в момент времени t , модель предсказывает распределение вероятностей $p^{(t)} = p(\cdot|y_1, \dots, y_{t-1}, x_1, \dots, x_m)$. Цель на этом этапе $-p^* = \text{one-hot}(y_t)$, т.е. мы хотим, чтобы модель присваивала вероятность 1 правильному токену, y_t , а остальным — ноль.

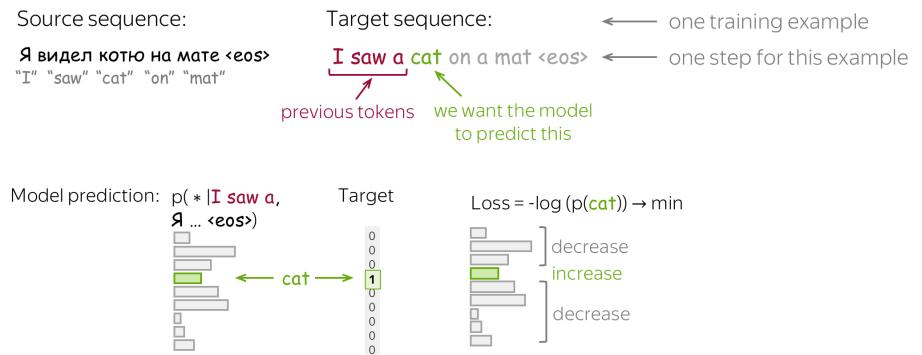
Стандартная функция потерь — это кросс-энтропийная потеря . Кросс-энтропийная потеря для целевого распределения p^* и предсказанное распределение p является

$$\text{Loss}(p^*, p) = -p^* \log(p) = -\sum_{i=1}^{|V|} p_i^* \log(p_i).$$

Поскольку только один из p_i^* не равен нулю (для правильного токена y_t), мы получим

$$\text{Loss}(p^*, p) = -\log(p_{y_t}) = -\log(p(y_t|y_{<t}, x)).$$

На каждом шаге мы максимизируем вероятность, которую модель присваивает правильному токену. Посмотрите на иллюстрацию для одного временного шага.



Для всего примера убыток составит $-\sum_{t=1}^n \log(p(y_t|y_{<t}, x))$. Посмотрите на иллюстрацию процесса обучения (иллюстрация для модели RNN, но модель может быть другой).

**Seq2seq и внимание**

Основы Seq2seq

Внимание

Трансформатор

Сегментация подслов: BPE

Анализ и интерпретируемость

Исследовательское мышление

Связанные статьи

Веселиться!

Encoder: read source



we are here
 Source: Я видел котю на мате <eos>
 "I" "saw" "cat" "on" "mat"
 Target: I saw a cat on a mat <eos>

Вывод: жадное декодирование и поиск луча

Теперь, когда мы понимаем, как может выглядеть модель и как её обучить, давайте подумаем, как сгенерировать перевод с её помощью. Мы моделируем вероятность предложения следующим образом:

$$y' = \arg \max_y p(y|x) = \arg \max_y \prod_{t=1}^n p(y_t|y_{<t}, x) \quad \text{How to find the argmax?}$$

Теперь главный вопрос: как найти argmax?

Обратите внимание, что мы не можем найти точное решение. Общее количество гипотез, которые нам нужно проверить, равно $|V|^n$, что на практике не осуществимо. Поэтому найдём приближённое решение.

Лена: На самом деле точное решение обычно хуже приближенных, которые мы будем использовать.

- Жадное декодирование: на каждом шаге выбирайте наиболее вероятный токен

Простая стратегия декодирования является жадной: на каждом шаге генерируется токен с наибольшей вероятностью. Это может быть хорошей отправной точкой, но этот метод изначально имеет изъян: лучший токен на текущем шаге не обязательно приводит к лучшей последовательности.

$$\arg \max_y \prod_{t=1}^n p(y_t|y_{<t}, x) \neq \prod_{t=1}^n \arg \max_{y_t} p(y_t|y_{<t}, x)$$

- Поиск луча: отслеживайте несколько наиболее вероятных гипотез

Вместо этого давайте сохраним несколько гипотез. На каждом шаге мы будем продолжать каждую из текущих гипотез и выбирать первые N из них. Это называется лучевым поиском .

<bos>

Start with the begin of sentence token or with an empty sequence

Обычно размер пучка составляет от 4 до 10. Увеличение размера пучка неэффективно с точки зрения вычислений и, что ещё важнее, приводит к ухудшению качества.

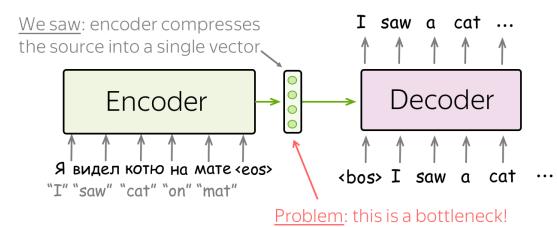


Внимание

Проблема представления фиксированного кодера

Проблема : фиксированное представление источника не является оптимальным: (i) для кодера сложно сжать предложение; (ii) для декодера на разных этапах может быть важна разная информация.

В рассмотренных нами моделях кодер сжимал всё исходное предложение в один вектор. Это может быть очень сложно, поскольку количество возможных исходных предложений (а значит, и их значений) бесконечно. Когда кодер вынужден сводить всю информацию в один вектор, он, скорее всего, что-то забудет.



Лена : Представьте себе всю Вселенную во всей её красоте — постараитесь визуализировать всё, что там есть, и как это можно описать словами. Затем представьте, что всё это скжато в один вектор размером, например, 512. Чувствуете ли вы, что со Вселенной всё в порядке?

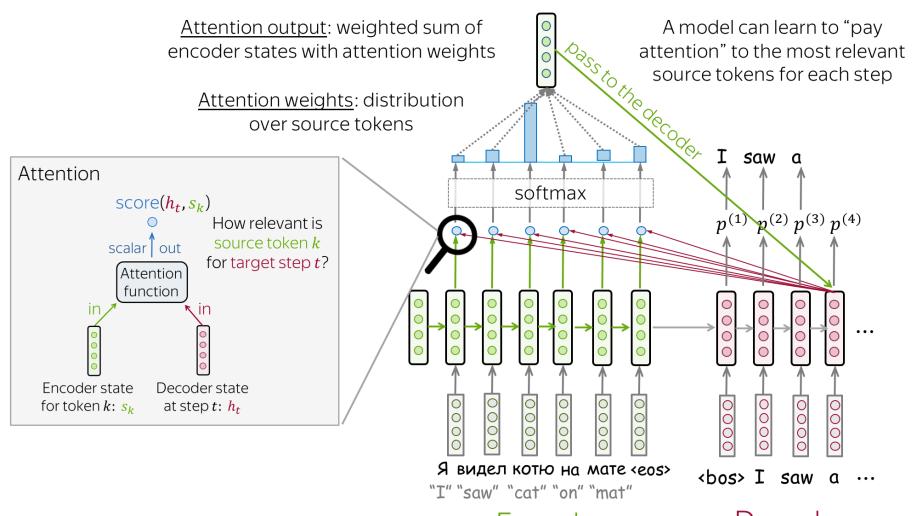
Свести всю информацию в один вектор сложно не только для кодера, но и для декодера. Декодер видит только одно представление исходного файла. Однако на каждом этапе генерации разные части исходного файла могут быть полезнее других. В текущих условиях декодеру приходится извлекать необходимую информацию из одного и того же фиксированного представления, что не так-то просто.

Внимание: общий вид

В статье «[Нейронный машинный перевод](#)» от [Jointly Learning to Align and Translate](#) было уделено внимание решению проблемы фиксированного представления.

Внимание : на разных этапах позволяйте модели «фокусироваться» на разных частях входных данных.

Механизм внимания является частью нейронной сети. На каждом шаге декодера он решает, какие части исходного текста наиболее важны. В этом случае кодеру не нужно сжимать весь исходный текст в один вектор — он предоставляет представления для всех исходных токенов (например, все состояния RNN вместо последнего).



На каждом шаге декодера внимание



Seq2seq и внимание

Основы Seq2seq

Внимание

Трансформатор

Сегментация подслов: BPE

Анализ и интерпретируемость



Исследовательское мышление



Связанные статьи

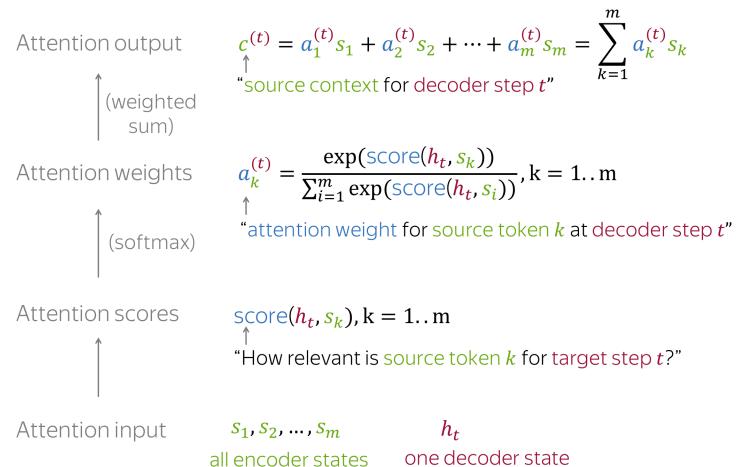


Веселиться!



- получает входной сигнал внимания : состояние декодера h_t и все состояния кодировщика s_1, s_2, \dots, s_m ;
- вычисляет баллы внимания для каждого состояния кодировщика s_k , внимание вычисляет свою «релевантность» для этого состояния декодера h_t . Формально, он применяет функцию внимания, которая получает одно состояние декодера и одно состояние энкодера и возвращает скалярное значение $score(h_t, s_k)$;
- вычисляет веса внимания : распределение вероятностей - softmax, применяемое к оценкам внимания;
- вычисляет выход внимания : взвешенную сумму состояний кодировщика с весами внимания.

Общая схема вычислений представлена ниже.

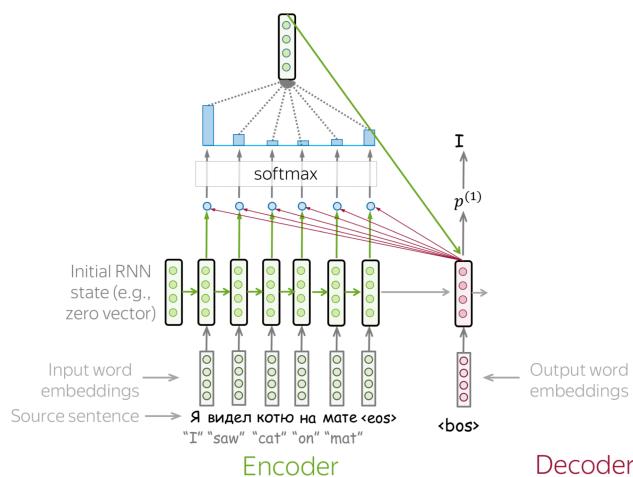


Примечание: Все дифференцируемо - изучается от начала до конца!

Основная идея заключается в том, что сеть может запоминать, какие входные данные наиболее важны на каждом этапе. Поскольку всё здесь дифференцируемо (функция внимания, softmax и всё остальное), модель с вниманием может быть обучена сквозным образом. Вам не нужно специально обучать модель выбирать нужные слова — она сама научится выбирать важную информацию .



Как: просматривайте слайды в удобном для вас темпе. Обратите внимание, как меняется вес внимания от шага к шагу — какие слова наиболее важны на каждом этапе?



- 1.
- 2.
- 3.
- 4.
- 5.

6.
7.
8.**Seq2seq и внимание**

Основы Seq2seq



Внимание



Трансформатор



Сегментация подслов: ВРЕ

Анализ и интерпретируемость



Исследовательское мышление



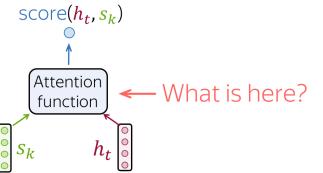
Связанные статьи



Веселиться!

**Как вычислить показатель внимания?**

В общей схеме, описанной выше, мы не указали, как именно мы вычисляем показатели внимания. Вы можете использовать любую функцию, даже очень сложную. Однако обычно это не требуется — существует несколько популярных и простых вариантов, которые работают довольно хорошо.



Dot-product

$$\begin{matrix} h_t^T \\ \text{---} \end{matrix} \times \begin{matrix} S_k \\ \text{---} \end{matrix}$$

Bilinear

$$\begin{matrix} h_t^T \\ \text{---} \end{matrix} \times \begin{matrix} W \\ \text{---} \end{matrix} \times \begin{matrix} S_k \\ \text{---} \end{matrix}$$

Multi-Layer Perceptron

$$\begin{matrix} w_2^T \\ \text{---} \end{matrix} \times \tanh \left(\begin{matrix} W_1 \\ \text{---} \end{matrix} \times \begin{matrix} h_t \\ \text{---} \end{matrix} \right) \times \begin{matrix} S_k \\ \text{---} \end{matrix}$$

$$\text{score}(h_t, s_k) = h_t^T s_k \quad \text{score}(h_t, s_k) = h_t^T W s_k \quad \text{score}(h_t, s_k) = w_2^T \cdot \tanh(W_1[h_t, s_k])$$

Наиболее популярные способы расчета баллов внимания:

- скалярное произведение - простейший метод;
- билинейная функция (также известная как «внимание Луонга») — используется в статье « Эффективные подходы к нейронному машинному переводу на основе внимания » ;
- многослойный персептрон (он же «внимание Багданау») — метод, предложенный в оригинальной статье .

Варианты модели: Бахданау и Луонг

Говоря о ранних моделях внимания, вы, скорее всего, услышите следующие варианты:

- Внимание Багданау - из статьи « Нейронный машинный перевод с помощью совместного обучения выравниванию и переводу» Дмитрия Багданау, Кьюнг-Хьюона Чо и Йошуа Бенджио (в этой статье впервые был представлен механизм внимания);
- Внимание Луонга - из статьи « Эффективные подходы к нейронному машинному переводу на основе внимания» Минь-Хянг Луонг, Хиен Фам, Кристофер Д. Мэннинг.

Они могут относиться к функциям оценки всех моделей, используемых в этих работах. В этой части мы подробнее рассмотрим эти два варианта моделей.

Модель Багданау

- Кодер: двунаправленный.

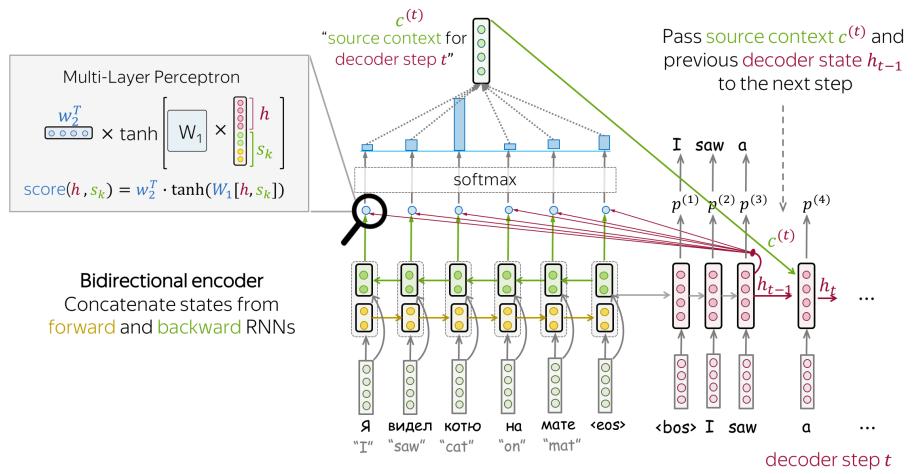
Для лучшего кодирования каждого исходного слова кодер использует две рекуррентные нейронные сети (РНС), прямую и обратную, которые считывают входные данные в противоположных направлениях. Для каждого токена состояния двух РНС объединяются.

- Оценка внимания: многослойный персептрон

Чтобы получить оценку внимания, примените многослойный персептрон (MLP) к состоянию кодера и состоянию декодера.

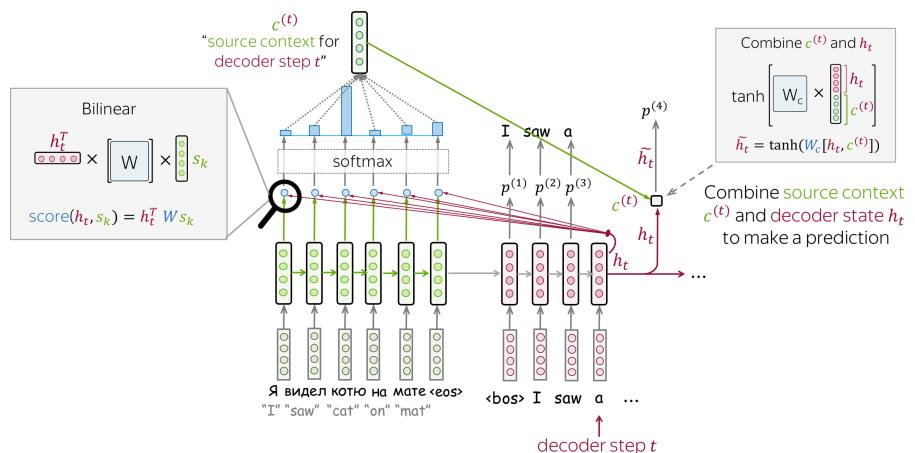
- внимание применяется: между шагами декодера

Внимание используется между шагами декодера: состояние h_{t-1} используется для вычисления внимания и его выходных данных $c^{(t)}$, и оба h_{t-1} и $c^{(t)}$ передаются в декодер на шаг t .

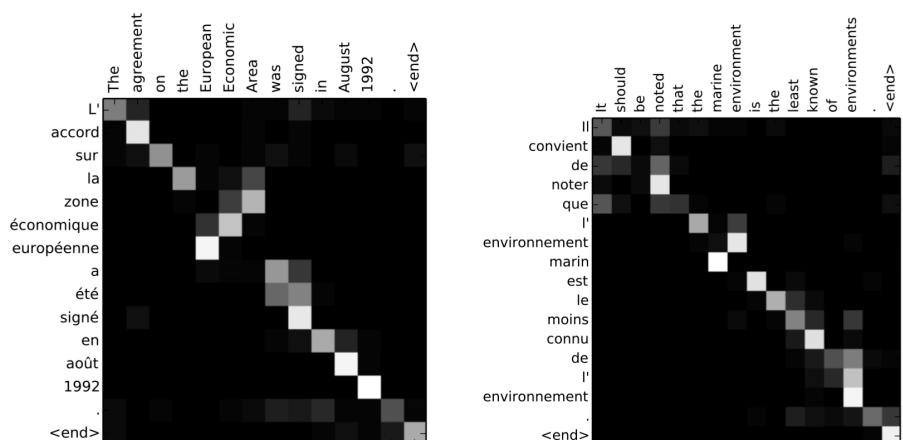
Модель Луонга

Хотя в [статье](#) рассматриваются несколько вариантов модели, та, которая обычно называется «внимание Луонга», выглядит следующим образом:

- кодер: односторонний (простой)
 - оценка внимания: линейная функция
 - внимание: между состоянием декодера RNN t и прогнозом для этого шага
- Внимание используется после шага декодера RNN t . Прежде чем сделать прогноз h_t , используется для вычисления внимания и его результатов $c(t)$. Затем h_t сочетается с $c(t)$ чтобы получить обновленное представление \tilde{h}_t , который используется для получения прогноза.

Внимание усваивает (почти) выравнивание

Помните о мотивации внимания? На разных этапах декодеру может потребоваться сосредоточиться на разных исходных токенах, которые более релевантны на данном этапе. Давайте рассмотрим весовые коэффициенты внимания — какие исходные слова использует декодер?





Из примеров мы видим, что внимание усвоило (мягкое) выравнивание между исходными и целевыми словами — декодер смотрит на те исходные токены, которые он переводит на текущем этапе.

Лена : «Выравнивание» — термин из статистического машинного перевода, но в этой части достаточно его интуитивного понимания как «что переводится во что».

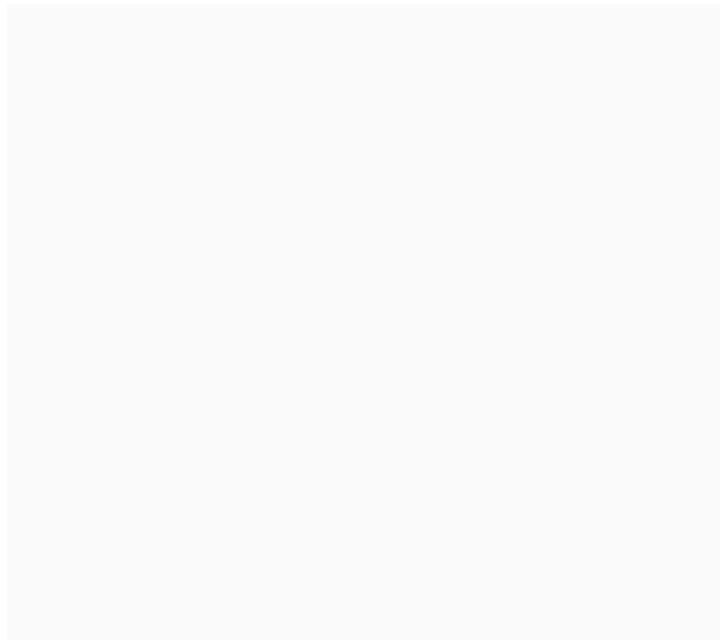
Трансформер: Внимание — это всё, что вам нужно

Трансформер — модель, представленная в статье «[Внимание — всё, что вам нужно](#)» в 2017 году. Она основана исключительно на механизмах внимания, то есть без рекуррентности и свёрток. Помимо более высокого качества перевода, модель обучается на порядок быстрее. В настоящее время трансформеры (с некоторыми вариациями) являются де-факто стандартными моделями не только для задач «последовательность в последовательность», но и для языкового моделирования и предобучения, что мы рассмотрим в следующей лекции.

Transformer представил новую парадигму моделирования: в отличие от предыдущих моделей, где обработка в кодере и декодере выполнялась с помощью рекуррентности или сверток, Transformer работает, используя только внимание.

	Seq2seq without attention	Seq2seq with attention	Transformer
processing within encoder	RNN/CNN	RNN/CNN	attention
processing within decoder	RNN/CNN	RNN/CNN	attention
decoder-encoder interaction	static fixed-sized vector	attention	attention

Посмотрите на иллюстрацию из блога [Google AI](#), представляющую Transformer.



Анимация взята из записи в блоге [Google AI](#).

Не вдаваясь в подробности, давайте выразим словами то, что мы только что увидели на иллюстрации. Получится что-то вроде следующего:



Encoder

Who is doing:

- all source tokens

What they are doing:

- look at each other
- update representations

Decoder

Who is doing:

- target token at the current step

What they are doing:

- looks at previous target tokens
- looks at source representations
- update representation

Хорошо, но есть ли причины, по которым это может быть более подходящим, чем RNN, для понимания языка? Давайте рассмотрим пример.

При кодировании предложения рекуррентные нейронные сети не поймут, что означает слово «bank», пока не прочитают его целиком, и для длинных последовательностей это может занять некоторое время. В отличие от этого, в кодировщике Transformer токены взаимодействуют друг с другом одновременно.

Интуитивно кодер Трансформера можно представить как последовательность этапов рассуждения (слоёв). На каждом этапе токены смотрят друг на друга (именно здесь нам требуется внимание — самовнимание), обмениваются информацией и пытаются лучше понять друг друга в контексте всего предложения. Это происходит на нескольких уровнях (например, 6).

В каждом слое декодера токены префикса также взаимодействуют друг с другом посредством механизма самовнимания, но, кроме того, они смотрят на состояния кодера (без этого перевод невозможен, верно?).

Теперь давайте попробуем понять, как именно это реализовано в модели.

Само-внимание : часть «Посмотрите друг на друга»

Внутреннее внимание — один из ключевых компонентов модели. Разница между вниманием и внутренним вниманием заключается в том, что внутреннее внимание действует между представлениями одной и той же природы: например, между всеми состояниями кодировщика в некотором слое.

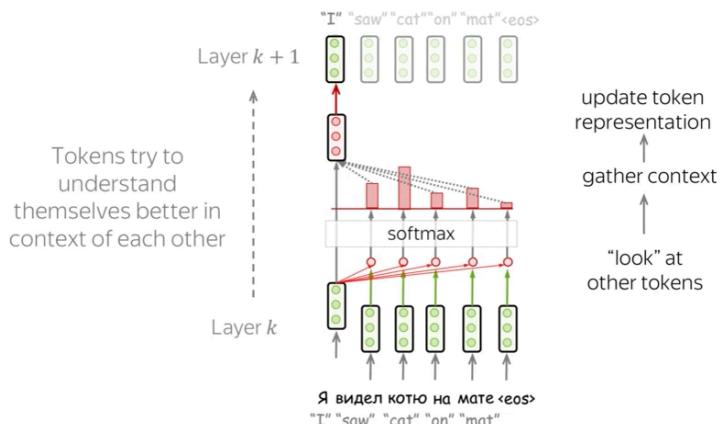
Decoder-encoder attention is looking

- from: one current decoder state
- at: all encoder states

Self-attention is looking

- from: each state from a set of states
- at: all other states in the same set

Самовосприятие — это часть модели, где токены взаимодействуют друг с другом. Каждый токен «смотрит» на другие токены в предложении с помощью механизма внимания, собирает контекст и обновляет предыдущее представление о себе. Взгляните на иллюстрацию.



Обратите внимание, что на практике это происходит параллельно.

Запрос, ключ и ценность во внимании к себе

Формально эта интуиция реализуется с помощью внимания типа «запрос-ключ-значение». Каждый входной токен в контексте внимания получает три представления, соответствующие ролям, которые он может играть:

- запрос - запрашивание информации;
- ключ - говорит о том, что в нем есть некоторая информация;
- ценность - предоставление информации.

Запрос используется, когда токен смотрит на других — он ищет информацию, чтобы лучше понять себя. Ключевым моментом является ответ на запрос: он используется для вычисления весов внимания. Значение используется для вычисления выходного внимания: оно предоставляет информацию токенам, которые «говорят», что им это нужно (т.е. присваивают этому токену большие веса).

Each vector receives three representations (“roles”)

$$[W_Q] \times [] = [] \quad \text{Query: vector from which the attention is looking}$$

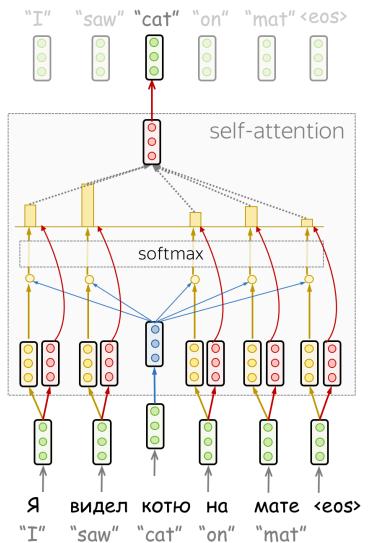
“Hey there, do you have this information?”

$$[W_K] \times [] = [] \quad \text{Key: vector at which the query looks to compute weights}$$

“Hi, I have this information – give me a large weight!”

$$[W_V] \times [] = [] \quad \text{Value: their weighted sum is attention output}$$

“Here’s the information I have!”



Формула для расчета выходного сигнала внимания выглядит следующим образом:

$$\text{Attention}(q, k, v) = \underbrace{\text{softmax}\left(\frac{qk^T}{\sqrt{d_k}}\right)}_{\text{Attention weights}} v$$

from to

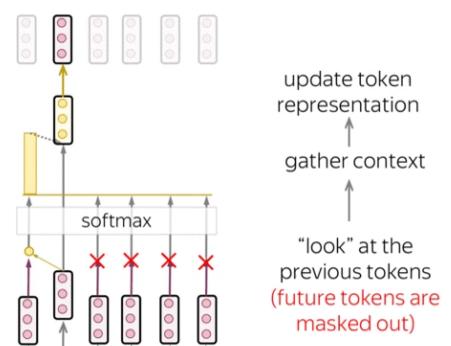
vector dimensionality of K, V

Скрытое внимание : «Не смотри вперёд» для декодера

В декодере также есть механизм внутреннего внимания: именно он выполняет функцию «посмотреть на предыдущие токены».

В декодере внутреннее внимание немного отличается от внимания кодера. В то время как кодер получает все токены сразу, и токены могут учитывать все токены во входном предложении, в декодоре мы генерируем по одному токену за раз: во время генерации мы не знаем, какие токены будут сгенерированы в будущем.

Чтобы запретить декодеру заглядывать вперёд, модель использует маскированное внутреннее внимание: будущие токены маскируются. Посмотрите на иллюстрацию.



Но как декодер может заглянуть вперед?

Во время генерации это невозможно — мы не знаем, что будет дальше. Но при обучении мы используем референтные переводы (которые нам известны). Поэтому при обучении мы передаем декодеру всё целевое предложение целиком — без масок токены «видели бы будущее», а это не то, что нам нужно.

Это сделано для повышения вычислительной эффективности: в Transformer нет рекуррентности, поэтому все токены могут быть обработаны одновременно. Это одна из



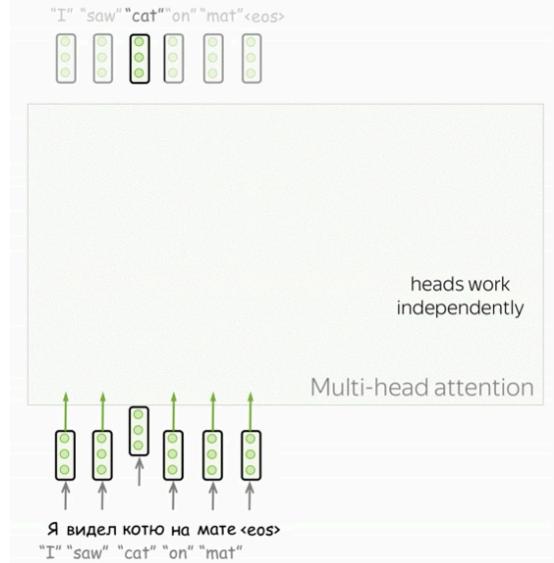
причин его популярности в машинном переводе — он обучается гораздо быстрее, чем некогда доминирующие рекуррентные модели. Для рекуррентных моделей один шаг обучения требует $O(\text{len(исходная модель)} + \text{len(целевая модель)})$ шагов, но для Transformer это $O(1)$, то есть константа.

Многоголовое внимание : независимая концентрация на разных вещах

Обычно понимание роли слова в предложении требует понимания того, как оно связано с различными его частями. Это важно не только для обработки исходного предложения, но и для создания целевого. Например, в некоторых языках подлежащие определяют склонение глагола (например, согласование рода), глаголы определяют падеж своих дополнений и многое другое. Я пытаюсь сказать следующее: каждое слово является частью множества отношений.

Следовательно, нам нужно позволить модели фокусироваться на разных объектах: именно в этом и заключается суть многоголового внимания. Вместо одного механизма внимания многоголовое внимание имеет несколько «голов», работающих независимо.

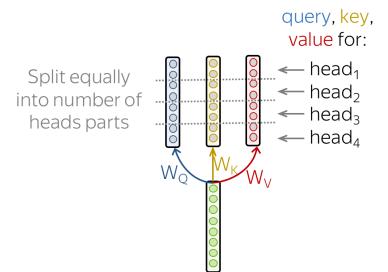
Формально это реализуется как несколько механизмов внимания, результаты которых суммируются:



$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_n)W_o,$$

$$\text{head}_i = \text{Attention}(QW_Q^i, KW_K^i, VW_V^i)$$

В этой реализации вы просто разделяете запросы, ключи и значения, вычисляемые для модели с одним центром внимания, на несколько частей. Таким образом, модели с одним или несколькими центрами внимания имеют одинаковый размер — многоцентровое внимание не увеличивает размер модели.



В разделе «Анализ и интерпретируемость» мы увидим, что эти заголовки играют разные «роли» в модели: например, позиционные или отслеживающие синтаксические зависимости.

Трансформер: Архитектура модели

Теперь, когда мы разобрались с основными компонентами модели и общей идеей, давайте рассмотрим её целиком. На рисунке показана архитектура модели из [оригинальной статьи](#).

Seq2seq и внимание

Основы Seq2seq

Внимание

Трансформатор

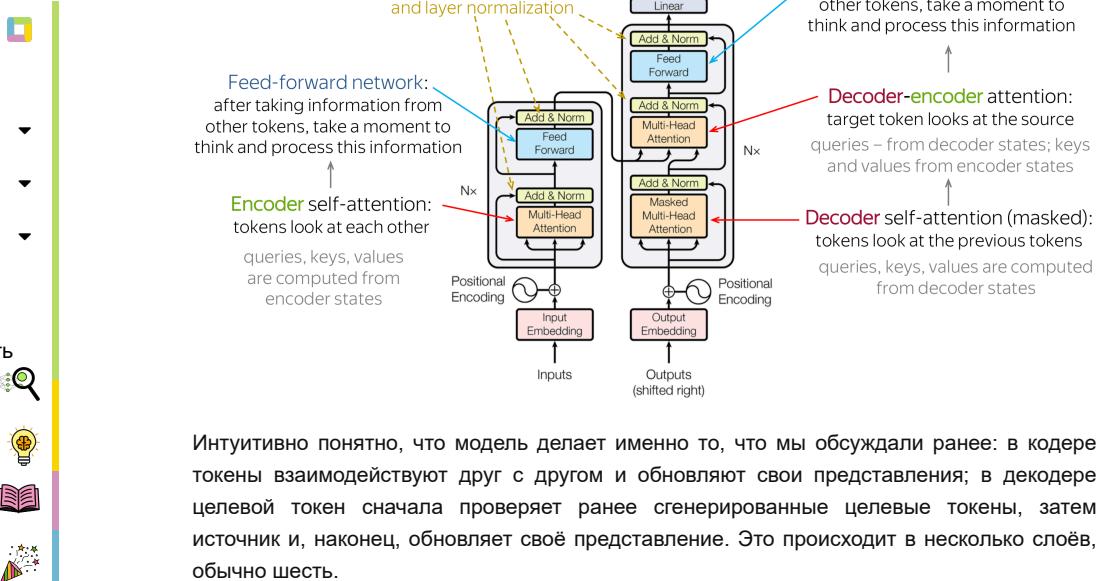
Сегментация подслов: BPE

Анализ и интерпретируемость

Исследовательское мышление

Связанные статьи

Веселиться!



Интуитивно понятно, что модель делает именно то, что мы обсуждали ранее: в кодере токены взаимодействуют друг с другом и обновляют свои представления; в декодере целевой токен сначала проверяет ранее сгенерированные целевые токены, затем источник и, наконец, обновляет своё представление. Это происходит в несколько слоёв, обычно шесть.

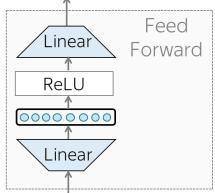
Давайте более подробно рассмотрим остальные компоненты модели.

- Блоки прямой связи

Помимо внимания, каждый слой имеет блок сети прямой связи: два линейных слоя с нелинейностью ReLU между ними:

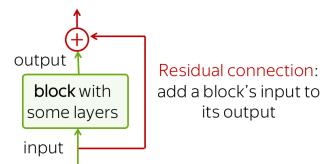
$$FFN(x) = \max(0, xW_1 + b_1)W_2 + b_2.$$

Посмотрев на другие токены с помощью механизма внимания, модель использует блок FFN для обработки этой новой информации (внимание — «посмотреть на другие токены и собрать информацию», FFN — «уделить немного времени размышлению и обработке этой информации»).



- Остаточные соединения

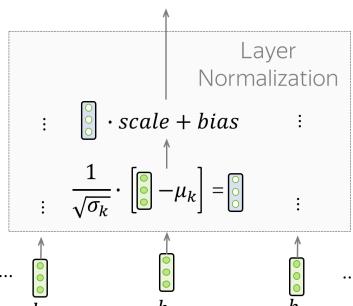
Мы уже рассматривали остаточные связи, когда говорили о [свёрточных языковых моделях](#). Остаточные связи очень просты (добавляют вход блока к его выходу), но в то же время очень полезны: они облегчают прохождение градиента через сеть и позволяют накладывать множество слоёв.



В Transformer остаточные связи используются после каждого блока «Внимание» и блока FFN. На иллюстрации выше остаточные связи показаны стрелками, огибающими блок и ведущими к ёлочному слою «Add & Norm». В блоке «Add & Norm» часть «Add» обозначает остаточную связь.

- Нормализация слоев

Часть «Norm» в слое «Add & Norm» обозначает [нормализацию слоя](#). Она независимо нормализует векторное представление каждого примера в пакете — это делается для управления «переходом» к следующему слову. Нормализация слоя улучшает стабильность сходимости, а иногда и качество.

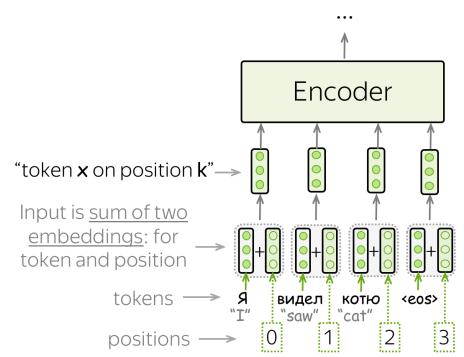


В Transformer необходимо нормализовать векторное представление каждого токена. Кроме того, LayerNorm имеет обучаемые параметры, $scale$ и $bias$, которые используются после нормализации для масштабирования выходных данных слоя (или входных данных следующего слоя). Обратите внимание, что μ_k и σ_k оцениваются для каждого примера, но $scale$ и $bias$ одинаковы — это параметры слоя.

- Позиционное кодирование



Обратите внимание, что, поскольку Transformer не содержит рекуррентных и свёрточных функций, он не знает порядок входных токенов. Поэтому нам необходимо явно указать модели их позиции. Для этого у нас есть два набора векторов: для токенов (как обычно) и для позиций (новые, необходимые для этой модели). Тогда входное представление токена будет суммой двух векторов: токенового и позиционного.



Позиционные вложения можно изучить, но авторы обнаружили, что наличие фиксированных вложений не влияет на качество. В Transformer используются следующие фиксированные позиционные кодировки:

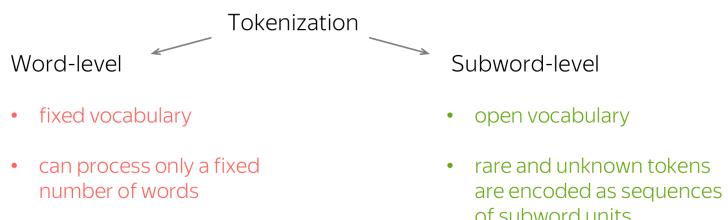
$$\text{PE}_{pos,2i} = \sin(pos/10000^{2i/d_{model}}),$$

$$\text{PE}_{pos,2i+1} = \cos(pos/10000^{2i/d_{model}}),$$

где pos — это положение i ; — размерность вектора. Каждое измерение позиционного кодирования соответствует синусоиде, а длины волн образуют геометрическую прогрессию от 2π до $10000 \cdot 2\pi$.

Сегментация подслов: кодирование пар байтов

Как мы знаем, модель имеет предопределённый словарь токенов. Входные токены, которых нет в словаре, будут заменены специальным токеном UNK («неизвестный»). Таким образом, если вы используете простую токенизацию на уровне слов (т.е. ваши токены — это слова), вы сможете обработать фиксированное количество слов. Это и есть проблема фиксированного словаря: вы получите много неизвестных токенов, и ваша модель не сможет их правильно перевести.



Но как представить все слова, даже те, которых мы не встречали в обучающих данных? Даже если вы не знакомы со словом, вы знакомы с его составными частями — подсловами (в худшем случае, символами). Тогда почему бы нам не разбить редкие и неизвестные слова на более мелкие части?

Именно это было предложено в статье «[Нейронный машинный перевод редких слов с использованием подсловных единиц](#)» Рико Сеннриха, Барри Хэддоу и Александры Бирч. Они ввели фактически стандартную сегментацию подслов — кодирование пар байтов (BPE). BPE сохраняет часто встречающиеся слова нетронутыми, а редкие и неизвестные разбивает на более мелкие известные части.

Как это работает?

Первоначальное [кодирование пар байтов \(BPE\)](#) (Gage, 1994) — это простой метод сжатия данных, который итеративно заменяет наиболее часто встречающуюся пару байтов в последовательности одним неиспользуемым байтом. То, что мы сейчас называем BPE, — это адаптация этого алгоритма для сегментации слов. Вместо объединения часто встречающихся пар байтов он объединяет символы или последовательности символов.

Алгоритм BPE состоит из двух частей:

- обучение — изучение «правил BPE», т. е. какие пары символов следует объединять;
- вывод — применение изученных правил для сегментации текста.

Давайте рассмотрим каждый из них более подробно.

Курс НЛП | Для вас

Seq2seq и внимание

Основы Seq2seq

Внимание

Трансформатор

Сегментация подслов: BPE

Анализ и интерпретируемость

Исследовательское мышление

Связанные статьи

Веселиться!



Обучение: изучение правил BPE

На этом этапе алгоритм строит таблицу слияния и словарь токенов. Начальный словарь состоит из символов и пустой таблицы слияния. На этом этапе каждое слово сегментируется в последовательность символов. После этого алгоритм выглядит следующим образом:

- подсчет пар символов: сколько раз каждая пара встречается вместе в обучающих данных;
- найти наиболее часто встречающуюся пару символов;
- объединить эту пару — добавить объединение в таблицу объединений, а новый токен — в словарь.

На практике алгоритм сначала подсчитывает, сколько раз каждое слово встречается в данных. Используя эту информацию, он может легче подсчитывать пары символов. Обратите внимание также, что токены не пересекают границы слов — всё происходит внутри слов.

Взгляните на иллюстрацию. Здесь я покажу вам учебный пример: предположим, что в обучающих данных мы встретили слово cat 4 раза, mat 5 раз, a mats , mate , ate , eat 2, 3, 3, 2 раза соответственно. Нам также нужно задать максимальное количество слияний; обычно оно составляет от 4 до 32 тысяч в зависимости от размера набора данных, но для нашего учебного примера установим его равным 5.

Words in the data:

Initial vocabulary:	word	count	Current merge table:
characters	c a t	4	(empty)
↓	m a t	5	
Split each word into characters	m a t s	2	
	m a t e	3	
	a t e	3	
	e a t	2	

Когда мы достигли максимального количества слияний, не все слова были объединены в один токен. Например, слово mats сегментируется на два токена: mat@@ s . Обратите внимание, что после сегментации мы добавляем специальные символы @@ , чтобы различать токены, представляющие целые слова, и токены, представляющие части слов. В нашем примере mat и mat@@ — разные токены.

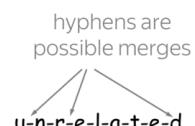
Примечание к реализации. При реализации необходимо убедиться, что новое слияние добавляет в словарь только один новый токен. Для этого можно либо добавить специальный символ конца слова к каждому слову (как это сделано в [оригинальной статье BPE](#)), либо заменить пробелы специальным символом (как это сделано, например, в [Sentencepiece](#) и [YouTokenToMe](#) , самой быстрой реализации), либо сделать что-то ещё. На рисунке я опустил это для простоты.

Вывод: сегментация текста

После изучения правил BPE у вас есть таблица слияния — теперь мы воспользуемся ею для сегментации нового текста.

Алгоритм начинается с сегментации слова на последовательность символов. После этого он итеративно выполняет следующие два шага, пока объединение не станет невозможным:

- среди всех возможных слияний на данном этапе найти наивысшее слияние в таблице;
- применить это слияние.



Обратите внимание, что таблица слияний упорядочена — слияния, расположенные выше в таблице, чаще встречались в данных. Поэтому в алгоритме слияния, расположенные выше, имеют

более высокий приоритет: на каждом шаге мы объединяем наиболее частое слияние среди всех возможных.

Курс НЛП | Для вас

Seq2seq и внимание

Основы Seq2seq

Внимание

Трансформатор

Сегментация подслов: BPE

Анализ и интерпретируемость

Исследовательское мышление

Связанные статьи

Веселиться!



Обратите внимание, что, хотя сегментация BPE детерминирована, даже при одинаковом словарном запасе слово может иметь разные сегментации, например, «не связан», «не связанный», «не связанный» и т. д.). Возможно, было бы лучше использовать разные сегментации при обучении?

Возможно, так и есть. Подробнее об этом [упражнении](#) в разделе «Исследовательское мышление».

🔍 Анализ и интерпретируемость

Многоголовое внутреннее внимание: что делают эти головы?

Начнём с нашего традиционного метода анализа модели: рассмотрим её компоненты. Ранее мы рассматривали свёрточные фильтры в классификаторах и нейроны в языковых моделях; теперь пришло время рассмотреть более крупный компонент: внимание. Но давайте возьмём не обычный, а головы из многоголовой модели внимания Трансформера.

Лена : Во-первых, зачем мы это делаем? Многоголовое внимание — это индуктивное смещение, представленное в Трансформере. Создавая индуктивное смещение в модели, мы обычно интуитивно понимаем, почему этот новый компонент модели, индуктивное смещение, может быть полезен. Поэтому полезно понять, как работает эта новая функция: учится ли она тому, что мы ожидали? Если нет, то почему это помогает? Если да, то как мы можем её улучшить? Надеюсь, теперь вы достаточно мотивированы, так что продолжим.

Наиболее важные заголовки поддаются интерпретации

Здесь мы упомянем некоторые результаты из статьи ACL 2019 года «[Анализ многоголового внимания: специализированные головы выполняют основную работу, остальное можно сократить](#)». Авторы рассматривают отдельные головы внимания в многоголовом анализе внимания кодировщика и оценивают, какой вклад в среднем вносят разные головы в генерируемые переводы (подробнее о том, как именно они это сделали, см. в статье или в [блоге](#)). Как оказалось,

- только небольшое количество голов имеет значение для перевода,
- эти головы играют интерпретируемые «роли».

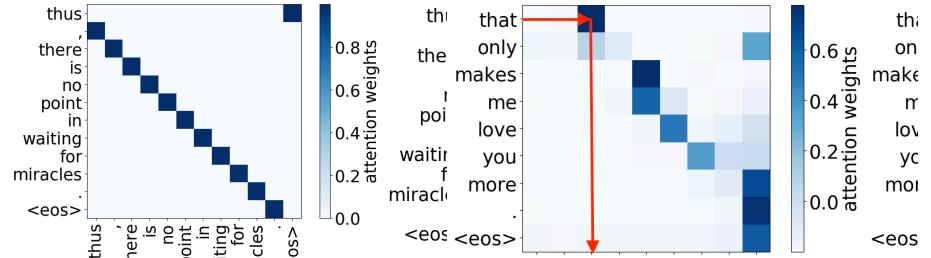
Эти роли таковы:

- позиционный : обращает внимание на ближайших соседей токена, и модель имеет несколько таких головок (обычно 2-3 головки, смотрящих на предыдущий токен, и 2 головки, смотрящих на следующий токен);
- синтаксический : научился отслеживать некоторые основные синтаксические отношения в предложении (подлежащее-сказуемое, глагол-дополнение);
- редкие токены : самая важная глава в первом слое обращает внимание на наименее частые токены в предложении (это справедливо для моделей, обученных на разных языковых парах!).

Взгляните на примеры позиционных и синтаксических головок ниже. Это означает, что наше интуитивное предположение о наличии нескольких головок оказалось верным — помимо прочего, модель научилась отслеживать связи между словами!

Позиционные головки

Синтаксические заголовки



Модель обучена на WMT EN-DE

Модель обучена на WMT EN-DE

Модель

- 1.
- 2.
- 3.
- 4.
- 5.
- 6.
- 7.
- 8.

Хотя заголовок с редкими токенами, безусловно, выглядит забавно, не стоит его переоценивать — скорее всего, это признак переобучения. Рассматривая наименее частые токены, модель пытается ухватиться за эти редкие «подсказки».

Большинство голов можно обрезать

Далее в статье авторы позволяют модели самостоятельно решать, какие из голов ей не нужны (опять же, более подробную информацию см. в статье или [блоге](#)), и итеративно удаляют из модели «головы внимания», то есть, удаляют их. Помимо подтверждения того, что специализированные головы являются наиболее важными (поскольку модель сохраняет их нетронутыми, а остальные удаляет), авторы обнаруживают, что большинство голов можно удалить без существенной потери качества.

Почему бы нам не обучить модель с небольшим количеством голов для начала?

Ну, это невозможно — качество будет гораздо ниже. Нужно много голов в обучении, чтобы они научились всем этим полезным вещам.

Зондирование: что отражают представления?

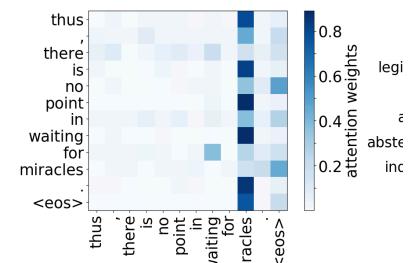
Обратите внимание, что рассмотрение компонентов модели является специфичным для модели подходом: компоненты, которые могут вас заинтересовать, и способы их «просмотра» зависят от модели.

Теперь нас интересуют методы, не зависящие от модели. Например, чему обучаются представления в модели? Обучаются ли они кодированию каких-либо лингвистических признаков? В данном случае мы передадим данные обученной сети, соберем векторные представления этих данных и попытаемся понять, кодируют ли эти векторы что-то интересное.

Самый популярный подход — использование зондирующих классификаторов (также известных как зонды, задачи зондирования, диагностические классификаторы). В этом случае вы

- передать данные в сеть и получить векторные представления этих данных,

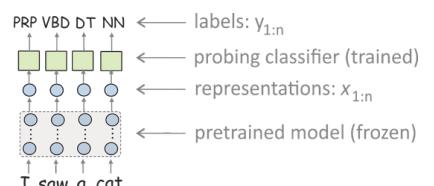
Редкие жетоны голова



Модель обучена на WMT EN-DE Mc



- 1.
- 2.
- 3.
- 4.
- 5.
- 6.
- 7.
- 8.
- 9.
- 10.
- 11.
- 12.
- 13.



**Seq2seq и внимание**

Основы Seq2seq

Внимание

Трансформатор

Сегментация подслов: ВРЕ

Анализ и интерпретируемость

Исследовательское мышление

Связанные статьи

Веселиться!

- обучить классификатор предсказывать некоторые лингвистические метки из этих представлений (но сама модель заморожена и используется только для создания представлений),
- использовать точность классификатора как меру того, насколько хорошо представления кодируют метки.

Этот подход к анализу (на данный момент) является самым популярным в NLP, и мы ещё встретимся с ним, когда будем говорить о переносе обучения на следующей лекции. А теперь давайте рассмотрим несколько примеров его применения для анализа моделей NMT.

Лена: Недавно выяснилось, что точность зондирующего классификатора — не самый лучший показатель, и нужно корректировать то, что вы оцениваете с помощью зондирующего классификатора. Но это совсем другая история...

Что модели NMT узнают о морфологии?

В этой части рассмотрим статью ACL 2017 года «[Что узнают о морфологии модели нейронного машинного перевода?](#)». Авторы обучили системы нейронного машинного перевода (LSTM) на нескольких языковых парах и проанализировали репрезентации этих моделей. Здесь я приведу лишь некоторые результаты, чтобы проиллюстрировать, как можно использовать зондирование для анализа.

- Теги частей речи

В одном из экспериментов изучается, насколько хорошо представления кодировщика улавливают теги частей речи (POS-теги). Для каждого из слоёв кодировщика, начиная с вложений, авторы обучили классификатор предсказывать POS-теги на основе представлений этого слова. Результаты показаны на рисунке (слой 0 — вложения, слои 1 и 2 — слои кодировщика).

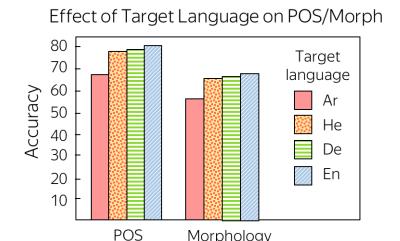
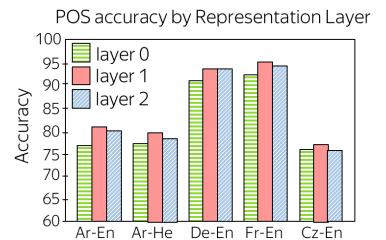
Мы можем видеть, что

- Прохождение вложений через кодировщик улучшает маркировку POS.
Это ожидаемо — в то время как слою 0 известен только текущий токен, представления со словами кодировщика знают его контекст и могут лучше понимать часть речи.
- Слой 1 лучше слоя 2
Гипотеза заключается в том, что слой 1 фиксирует структуру слова, а слой 2 кодирует более высокоуровневую информацию (например, семантическую).

- Влияние целевого языка

Ещё один интересный вопрос, который рассматривают авторы, — влияние целевого языка. При наличии одного и того же исходного языка (арабского) и разных целевых языков, кодировщики, обученные на каком целевом языке, лучше усвоят морфологию исходного текста? В качестве целевых языков авторы взяли арабский, иврит (язык с богатой морфологией, схожий с исходным языком), немецкий (язык с богатой морфологией, но с другой морфологией) и английский (язык с бедной морфологией).

Несколько неожиданно, более слабая морфология цели заставляет модель лучше понимать морфологию источника .

**Исследовательское мышление****Как**

- Прочитайте краткое описание в начале — это наша отправная точка, нечто известное.
- Прочитайте вопрос и подумайте: минуту, день, неделю... — дайте себе времени! Даже если вы не думаете об этом постоянно, что-то всё равно может прийти в



Seq2seq и внимание

Основы Seq2seq



Внимание



Трансформатор



Сегментация подслов: BPE

Анализ и интерпретируемость



Исследовательское мышление



Связанные статьи



Веселиться!



голову.

- Посмотрите на возможные ответы — предыдущие попытки решить эту задачу. Важно: не обязательно предлагать что-то в точности похожее — помните, каждая статья обычно занимает у авторов несколько месяцев. Важна привычка думать о таких вещах! Всё остальное, что нужно учёному, — это время: пробовать, ошибаться, думать, пока не получится.

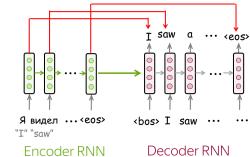
Известно, что изучение чего-либо проходит легче, если не сразу получить ответ, а сначала подумать над ним. Даже если вы не хотите становиться исследователем, это всё равно хороший способ учиться!

○ Простые LSTM-сети без внимания

Тренировочный трюк, позволяющий заставить работать простые LSTM-сети без привлечения внимания

Простые LSTM-сети без внимания работают не очень хорошо: все зависимости долгосрочные, и это сложно для модели. Например, к тому времени, как декодеру придётся генерировать начало перевода, он может уже забыть наиболее релевантные ранние исходные токены.

All dependencies are long-term!



? Можно ли изменить конвейер обучения (не изменяя модель), чтобы модели было проще запомнить начало источника, когда она начинает генерировать цель?

► Возможные ответы

○ Улучшить сегментацию подслов

Сделайте BPE стохастическим: сегментируйте слова по-разному

The standard BPE segmentation is deterministic: at each step, it always picks the highest merge in the table. However, even with the same vocabulary, a word can have different segmentations, e.g. un relat ed, u n relate d, un rel ated, etc.).

Imagine that during training of an NMT model each time we pick one of the several possible segmentations - i.e. the same word can be segmented differently. We use the same merge table built by the standard BPE, and the standard segmentation at test time - we modify only training.

BPE: deterministic

u-n-r-e-l-a-t-e-d
u-n re-l-a-t-e-d
u-n re-l-at-e-d
un re-l-at-ed
un re-l-ated
un rel-at-ed
un related

We want: stochastic

u-n-r-e-l-a-t-e-d
↓
un related ↓
u n relate d

? Do you think this would improve model quality? Why?

► Possible answers

? How would you change the segmentation procedure of BPE to enable different segmentations of the same word?

► Possible answers



Здесь будет больше упражнений!



Эта часть будет время от времени расширяться.



Связанные статьи

Как

- Высокий уровень : просмотрите основные результаты в кратких обзорах — получите представление о том, что происходит в данной области.
- Немного глубже : по темам, которые вас интересуют, читайте более подробные обзоры с иллюстрациями и пояснениями. Просмотрите ход рассуждений авторов и их ключевые наблюдения.
- Подробный анализ : прочтите понравившиеся статьи. Теперь, когда вы поняли основную идею, будет проще!

Здесь будут статьи!



Статьи будут появляться постепенно.



Веселиться!

Вскоре!



Мы все еще работаем над этим!



Передача обучения

Что такое трансферное обучение?

Обзор: Встраивание слов

Предварительно обученные модели

(Немного) адаптеров

(Заметка о) контрольных показателях

Анализ и интерпретируемость



Исследовательское мышление



Связанные статьи



Веселиться!



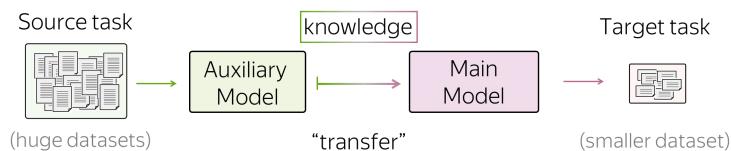
(Введение в) трансферное обучение

Лена : Тема трансферного обучения огромна, поэтому охватить её всю в одной лекции невозможно. Здесь я попытаюсь дать общее представление о трансфере и покажу несколько распространённых способов его реализации.

До эпохи больших языковых моделей (таких как ChatGPT) трансферное обучение, вероятно, было самой популярной областью обработки естественного языка как в исследованиях, так и в промышленности. Скорее всего, вы уже слышали разговоры об ELMo, BERT и других персонажах — после этой лекции вы поймёте, почему!

«Перенос» знаний из одной модели в другую

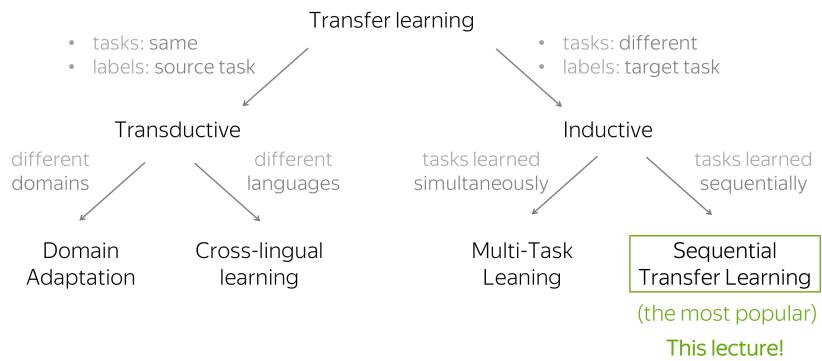
Общая идея трансферного обучения заключается в «переносе» знаний из одной задачи/модели в другую. Например, у вас нет большого объёма данных для интересующей вас задачи (например, классификации), и сложно построить хорошую модель, используя только эти данные. Вместо этого у вас могут быть данные для какой-то другой задачи, которую получить проще (например, для моделирования языка метки вообще не нужны — достаточно простого текста).



В этом случае вы можете «перенести» знания из задачи, которая вам не интересна (назовем ее исходной задачей), в задачу, которая вам интересна, т. е. в целевую задачу .

Таксономия переноса обучения в НЛП

Существует несколько типов переноса, которые подробно описаны в [блоге Себастьяна Рудера](#). Две большие категории — трансдуктивное и индуктивное трансферное обучение: они делят все подходы на те, где задача та же, а метки находятся только в исходном коде (трансдуктивные), и те, где задачи разные, а метки находятся только в целевом коде (индуктивные).



Эта таксономия взята из [записи в блоге Себастьяна Рудера](#).

В этой лекции нас интересует последняя категория с различными заданиями, а также ее подкатегория, которая изучает эти задания последовательно.

Лена : Обычно я неохотно делаю такие заявления, но здесь я считаю, что можно с уверенностью сказать, что последовательное трансферное обучение в настоящее время является одной из самых популярных областей исследований.

Что мы будем рассматривать

В этой лекции нас больше всего интересует то, как могут выглядеть вспомогательные модели и как выглядит сам трансфер со стороны моделирования.



Передача обучения

Что такое трансферное обучение?

Обзор: Встраивание слов

Предварительно обученные модели

(Немного) адаптеров

(Заметка о) контрольных показателях

Анализ и интерпретируемость



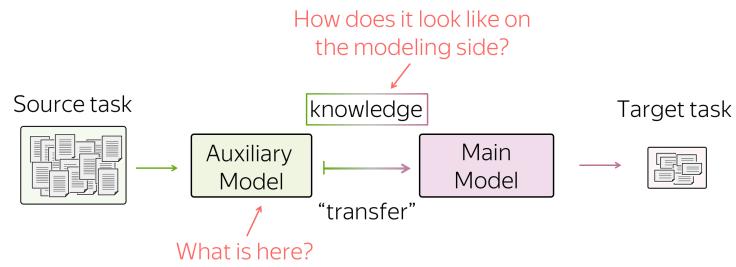
Исследовательское мышление



Связанные статьи



Веселиться!



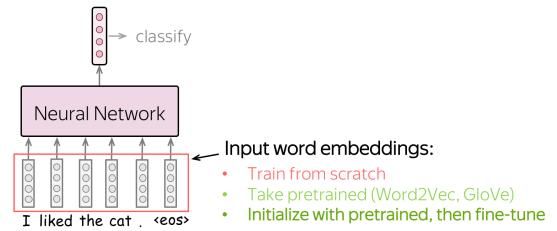
Простейший перенос: встраивание слов

Говоря о [классификации текстов](#), мы уже говорили, что использование предварительно обученных векторных представлений слов может быть очень полезным. Давайте повторим этот момент.

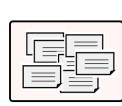
Резюме: Встраивание в классификацию текста

Входные данные для сети представлены в виде векторных представлений слов. Получить эти векторные представления для вашей модели можно тремя способами:

- обучение с нуля как часть вашей модели,
- взять предобученные (Word2Vec, GloVe и т.д.) и исправить их (использовать как статические векторы),
- инициализировать с помощью предварительно обученных вложений и обучите их с помощью сети («тонкая настройка»).

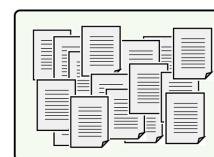


Давайте рассмотрим эти варианты, взглянув на данные, которые может использовать модель. Обучающие данные для классификации размечены и специфичны для конкретной задачи, но размеченные данные обычно сложно получить. Поэтому этот корпус, вероятно, будет не очень большим (как минимум), или не будет разнообразным, или и тем, и другим. Напротив, обучающие данные для векторных представлений слов не размечены — достаточно простых текстов. Следовательно, эти наборы данных могут быть огромными и разнообразными, что даёт массу возможностей для изучения.



Training data for text classification (labeled)

- Not huge, or not diverse, or both
- Domain: task-specific



Training data for word embeddings (unlabeled)

- Huge diverse corpus (e.g., Wikipedia)
- Domain: general

Теперь давайте подумаем о том, что будет знать модель в зависимости от того, что мы делаем с эмбеддингами. Если эмбеддинги обучаются с нуля, модель будет «знать» только данные классификации — этого может быть недостаточно для эффективного изучения связей между словами. Но если мы используем предобученные эмбеддинги, они (и, следовательно, вся модель) будут знать огромный корпус — они узнают много нового о мире. Чтобы адаптировать эти эмбеддинги к данным, специфичным для вашей задачи, вы можете точно настроить их, обучая всю сеть — это может дать прирост производительности (хотя и не слишком большой).

- Train from scratch

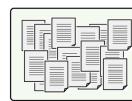
What they will know:



May be not enough to learn relationships between words

- Take pretrained (Word2Vec, GloVe)

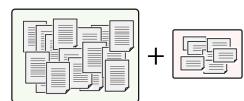
What they will know:



Know relationships between words, but are not specific to the task

- Initialize with pretrained, then fine-tune

What they will know:



Know relationships between words and adapted for the task

“Transfer” knowledge from a huge unlabeled corpus to your task-specific model

Использование предварительно обученных вложений является примером трансферного обучения : посредством вложений мы «переносим» знания их обучающих данных в нашу конкретную модель.

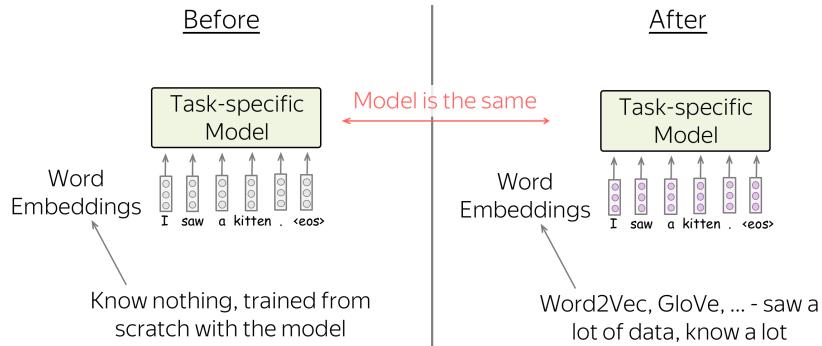


Передача посредством встраивания слов

Мы только что изложили основную идею использования предварительно обученных векторных представлений слов в моделях, ориентированных на конкретные задачи:

Посредством внедрения мы «переносим» знания из их обучающих данных в нашу специфичную для задачи модель.

В модели этот перенос реализуется посредством замены случайно инициализированных вложений на предобученные (что то же самое, копирование весов из предобученных вложений в вашу модель).



Обратите внимание, что мы не меняем модель : она остаётся точно такой же. Как мы увидим чуть позже, так будет не всегда.

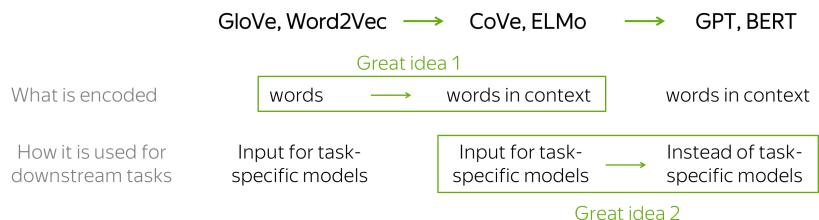
Предварительно обученные модели

Идея переноса знаний, сформулированная нами для векторных представлений, носит общий характер и остаётся неизменной при переходе от векторных представлений слов к предобученным моделям. То есть, буквально: вы можете просто заменить «векторные представления слов» на название вашей модели!

С помощью `_insert_your_model_` мы «переносим» знания из ее обучающих данных в модель, специфичную для задачи.

Две великие идеи

В этой части мы рассмотрим четыре модели: CoVe, ELMo, GPT и BERT. Обратите внимание, что сейчас существует множество вариаций этих моделей: от совсем небольших изменений (например, изменение обучающих данных и/или настроек) до весьма существенных (например, изменение целей обучения). Однако, грубо говоря, переход от векторных представлений слов к современным моделям можно объяснить всего двумя идеями.



Две замечательные идеи:

- что кодируется : от слов к словам в контексте (переход от Word2Vec/GloVe и т.д. к CoVe/ELMo);
- использование для задач нисходящего потока : от замены только вложенных слов в моделях, ориентированных на задачи, до замены целых моделей, ориентированных на задачи (переход от CoVe/ELMo к GPT/BERT).

Теперь я объясню каждую из этих идей вместе с соответствующими моделями.

Отличная идея 1 : от слов к словам в контексте



Передача обучения

Что такое трансферное обучение?

Обзор: Встраивание слов

Предварительно обученные модели

(Немного) адаптеров

(Заметка о) контрольных показателях

Анализ и интерпретируемость



Исследовательское мышление



Связанные статьи



Веселиться!



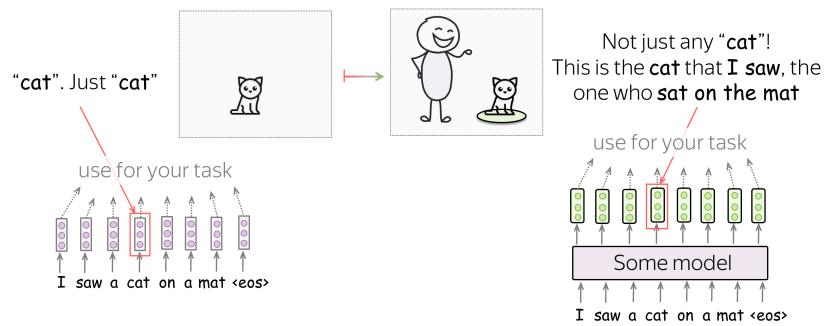
Как мы только что видели, передача знаний посредством векторных представлений существовала задолго до появления предобученных моделей: в простейшем случае — посредством словесных представлений. Гораздо более эффективной (и, следовательно, популярной) её сделала очень простая идея:

Вместо того чтобы представлять отдельные слова, мы можем научиться представлять слова вместе с контекстом, в котором они используются.

Ну, хорошо. Но как это сделать?

Помните, мы обучали нейронные языковые модели? Для обучения такой модели нужны те же данные, что и для обучения векторных представлений слов: простые тексты на естественном языке (например, тексты из Википедии, да что угодно). Обратите внимание, что для этого вам не нужны никакие метки!

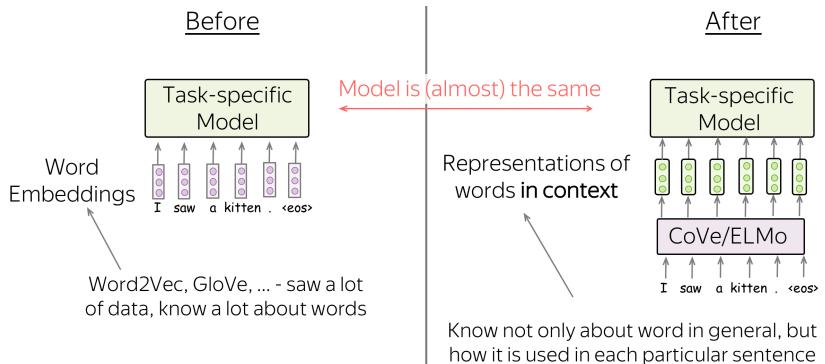
Представьте себе, что у нас есть тексты на естественном языке. Мы можем использовать их для обучения векторных представлений слов (Word2Vec, GloVe и т. д.) или нейронной языковой модели. Но обучение языковой модели дает гораздо больше, чем обучение векторных представлений слов: языковые модели обрабатывают не только отдельные слова, но и предложения/абзацы и т. д. Внутри модели LM также строят векторные представления для каждого слова, но эти векторы представляют не просто слова, а слова в контексте.



Например, рассмотрим предложение « Я видел кошку на коврике» и слово «кошка» в нём. Если мы используем векторные представления слов, вектор для слова «кошка» будет содержать информацию об общем понятии « кошка» : это может быть любая кошка, которую вы можете себе представить. Но если мы возьмём вектор для слова « кошка» откуда-то из языковой модели, это уже будет не кошка ! Поскольку языковые модуличитывают контекст, это векторное представление для слова «кошка» будет знать, что это та самая кошка , которую я видел , та, которая сидела на коврике .

Перевод : вместо вложений слов используйте представления

Мы рассмотрим две модели, которые первыми реализовали идею кодирования слов с учётом контекста: **CoVe** и **ELMo**. Их представления используются для последующих задач практически так же, как и в случае с векторными представлениями слов: обычно достаточно просто подставить представления вместо векторных представлений слов (там, где раньше находился, например, GloVe). Вот и всё!



Обратите внимание, что здесь по-прежнему используется модель, специфичная для каждой задачи, и эти модели могут существенно различаться. Изменился способ кодирования слов перед их передачей этим моделям.



Передача обучения

Что такое трансферное обучение?

Обзор: Встраивание слов

Предварительно обученные модели

(Немного) адаптеров

(Заметка о) контрольных показателях

Анализ и интерпретируемость



Исследовательское мышление



Связанные статьи



Веселиться!



Лена : В оригинальных статьях авторы предлагают некоторые модификации для моделей, ориентированных на конкретные задачи. Однако они довольно незначительны, и, грубо говоря, ими можно пренебречь. Важно то, что вместо представления отдельных слов CoVe и ELMo представляют слова в контексте.

Теперь осталось только указать

- это какая-то модель с иллюстрацией,
- какие представления следует взять из этой модели.

CoVe : Контекстуализированные векторы слов, изученные при переводе

CoVe расшифровывается как «векторы контекста» и был представлен в статье NeurIPS 2017 года « [Изученные при переводе: контекстуализированные векторы слов](#) ». Авторы впервые предложили научиться кодировать не только отдельные слова, но и слова вместе с их контекстом.

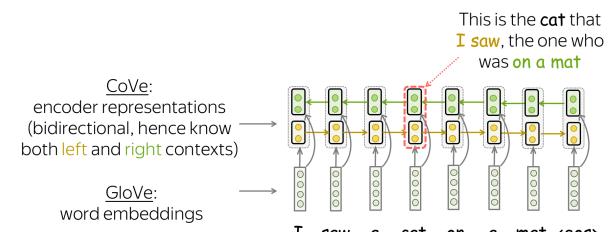
Обучение модели : нейронный машинный перевод (LSTM и внимание)

Для кодирования слов в контексте предложения/абзаца CoVe обучает систему НМП и использует её кодер. Основная гипотеза заключается в том, что для перевода предложения НМП-кодеры обучаются «понимать» исходное предложение. Следовательно, векторные представления, создаваемые кодером, содержат информацию о контексте слова.

Формально авторы обучают модель перевода LSTM с вниманием (например, модель Bahdanau, которую мы рассматривали в предыдущей лекции). Поскольку в конечном итоге мы хотим использовать обученный кодер для обработки предложений на английском языке (не потому, что нас интересует только английский, а потому, что большинство наборов данных для последующих задач на английском), система NMT должна выполнять перевод с английского на какой-либо другой язык (например, немецкий).

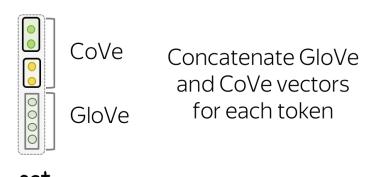
Двунаправленный кодер : знает как левый, так и правый контекст

Обратите внимание, что в этой модели NMT кодер работает в двух направлениях: он объединяет выходные данные прямой и обратной LSTM-сетей. Таким образом, выходные данные кодера содержат информацию как о левом, так и о правом контексте токена.



Получение представлений : объединение векторов GloVe и CoVe

После обучения модели NMT нам нужен только её кодер. Для заданного текста выходными данными кодера являются векторы CoVe. Для последующих задач авторы предлагают использовать конкатенацию векторов Glove (представляющих отдельные токены) и CoVe (токены, закодированные в контексте). Идея заключается в том, что эти векторы кодируют различные виды информации, и их комбинация может быть полезна.



Результаты : улучшения очевидны

Используя векторы CoVe вместе с GloVe, авторы добились значительных улучшений во многих последующих задачах: классификации текста, выводе на естественном языке и ответах на вопросы.



Передача обучения

Что такое трансферное обучение?

Обзор: Встраивание слов

Предварительно обученные модели

(Немного) адаптеров

(Заметка о) контрольных показателях

Анализ и интерпретируемость



Исследовательское мышление



Связанные статьи



Веселиться!

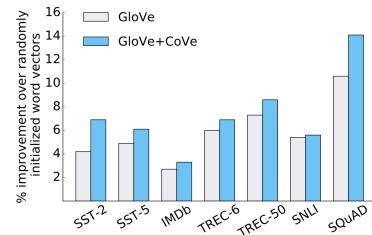


Рисунок взят из оригинальной статьи CoVe .

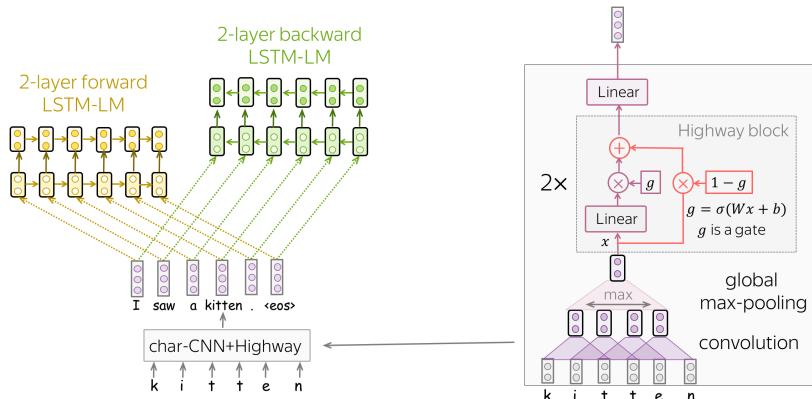
ELMo : Встраивание из языковых моделей

Модель ELMo была представлена в статье « Глубокие контекстуализированные представления слов ». В отличие от CoVe, ELMo использует представления не из модели NMT, а из языковой модели. Просто заменив вложения слова (GloVe) вложениями из LM, они получили значительное улучшение в решении ряда задач, таких как ответы на вопросы, разрешение крефераентов, анализ тональности, распознавание именованных сущностей и других. И, кстати, статья получила награду за лучшую статью на конференции NAACL 2018!

Теперь давайте рассмотрим ELMo подробнее.

Обучение модели : прямые и обратные LSTM-LM на основе char-CNN

Модель очень проста и состоит из двухслойных языковых моделей LSTM: прямой и обратной. Обе модели используются для того, чтобы каждый токен мог иметь оба контекста: левый и правый.



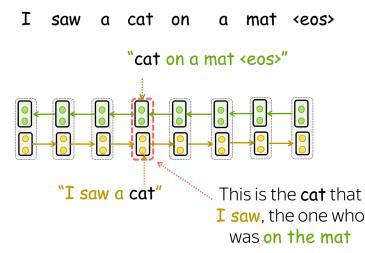
Интересно также, как авторы получают исходные представления слов (которые затем передаются в LSTM-сети). Напомним, что в стандартном слое встраивания слов для каждого слова в словаре мы обучаем уникальный вектор. В данном случае

- Вложения слова не знают символы, из которых они состоят (например, они не знают, что слово представлять , представляет , представленный и представление близки по написанию)
- мы не можем представлять слова, не входящие в словарный запас (OOV).

Чтобы решить эти проблемы, авторы представляют слова как выходные данные сети на уровне символов. Как видно из иллюстрации, эта сверточная нейронная сеть очень проста и состоит из компонентов, которые мы уже рассматривали ранее: свертки, глобального пула, магистральных связей и линейных слоёв. Таким образом, представления слов распознают свои символы по построению, и мы можем представить даже те слова, которые никогда не встречались нам в процессе обучения.

Получение представлений : весовые представления из разных слоев

После обучения модели мы можем использовать её для получения представлений слов. Для этого для каждого слова мы объединяем представления из соответствующих слоёв прямой и обратной LSTM-сетей. Объединяя эти прямые и обратные векторы, мы создаём представление слова, которое «знает» как о левом, так и о правом контекстах.





Передача обучения

Что такое трансферное обучение?

Обзор: Встраивание слов

Предварительно обученные модели

(Немного) адаптеров

(Заметка о) контрольных показателях

Анализ и интерпретируемость



Исследовательское мышление



Связанные статьи

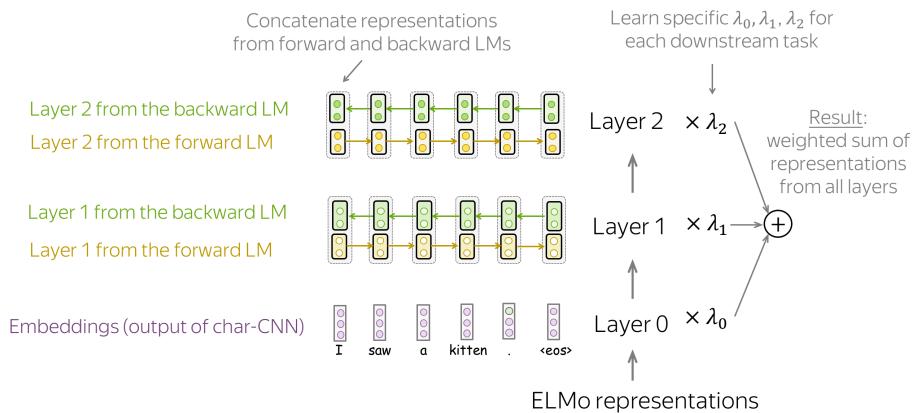


Веселиться!



В целом представления ELMo имеют три слоя:

- слой 0 (внедрения) — выходные данные CNN символьного уровня;
- слой 1 — объединенные представления слоя 1 как прямой, так и обратной LSTM;
- слой 2 — объединенные представления слоя 2 как прямой, так и обратной LSTM.



Слои содержат различную информацию → объедините их

Каждый из этих слоёв кодирует разные виды информации: слой 0 — только на уровне слов, слои 1 и 2 — слова в контексте. Сравнивая слои 1 и 2, слой 2, вероятно, содержит более высокоуровневую информацию: эти представления исходят из более высоких слоёв соответствующих языковых модулей.

Поскольку для разных задач нижестоящего уровня требуется разная информация, ELMo использует специфические для каждой задачи веса для объединения представлений из трёх слоёв. Это скаляры, которые изучаются для каждой задачи нижестоящего уровня. Результатирующий вектор, взвешенная сумма представлений из всех слоёв, используется для представления слова.

Отличная идея 2 : откажитесь от моделей, ориентированных на конкретные задачи

Следующие два (класса) моделей, которые мы рассмотрим, — это GPT и BERT, которые существенно отличаются от предыдущих подходов в том, как они используются для последующих задач:

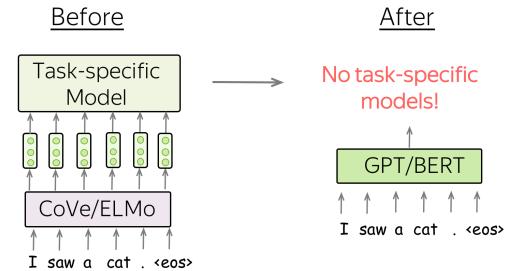
CoVe/ELMo заменяет встроенные слова, а GPT/BERT заменяют целые модели .

До : конкретная архитектура модели для каждой последующей задачи

Обратите внимание, что представления ELMo/CoVe в основном использовались для замены слоя встраивания, сохраняя архитектуру моделей, специфичную для конкретной задачи, практически нетронутой. Это означает, например, что для разрешения когерентности приходилось использовать специальную модель, разработанную для этой задачи, для разметки частей речи — другую модель, для вопросно-ответной — ещё одну, весьма специфичную модель и т. д. Для каждой из этих задач исследователи, специализирующиеся на ней, постоянно совершенствовали архитектуру моделей, специфичную для конкретной задачи.

После : Единая модель, которую можно точно настроить под все задачи

В отличие от предыдущих моделей, GPT/BERT заменяет не векторное представление слов, а модели, ориентированные на конкретные задачи. В этой новой модели модель сначала проходит предобучение на большом объёме немаркированных данных (обычных текстов). Затем эта модель настраивается под каждую из последующих задач. Важно, что теперь во время настройки нужно использовать только преобразования входных данных,



учитывающие специфику задачи (т.е. подавать данные определённым образом), а не изменять архитектуру модели.

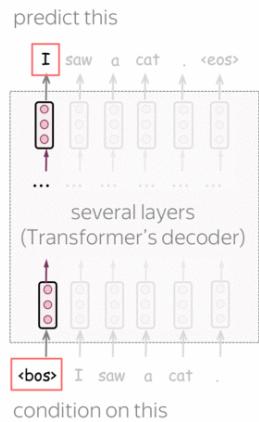
GPT : Генеративная предварительная подготовка к пониманию языка

Предварительное обучение: языковая модель «трансформер слева направо»

GPT — это языковая модель с написанием текста слева направо, основанная на Transformer. Архитектура представляет собой 12-слойный декодер Transformer (без участия декодера-кодера).

Формально, если y_1, \dots, y_n является последовательностью обучающих токенов, тогда на шаге времени t модель предсказывает распределение вероятностей $p^{(t)} = p(\cdot | y_1, \dots, y_{t-1})$. Модель обучается со стандартной кросс-энтропийной потерей, а потеря для всей последовательности составляет

$$L_{xent} = - \sum_{t=1}^n \log(p(y_t | y_{<t})).$$



Лена : Более подробную информацию о моделировании языка слева направо и/или модели Transformer можно найти в лекциях «[Моделирование языка](#)» и «[Seq2seq и внимание](#)» .

Тонкая настройка: использование GPT для нисходящих задач

Потери при тонкой настройке состоят из потерь, связанных с конкретной задачей, а также потерь при моделировании языка:

$$L = L_{xent} + \lambda \cdot L_{task}.$$

На этапе тонкой настройки архитектура модели остаётся прежней, за исключением последнего линейного слоя. Меняется лишь формат входных данных для каждой задачи: см. иллюстрацию ниже.

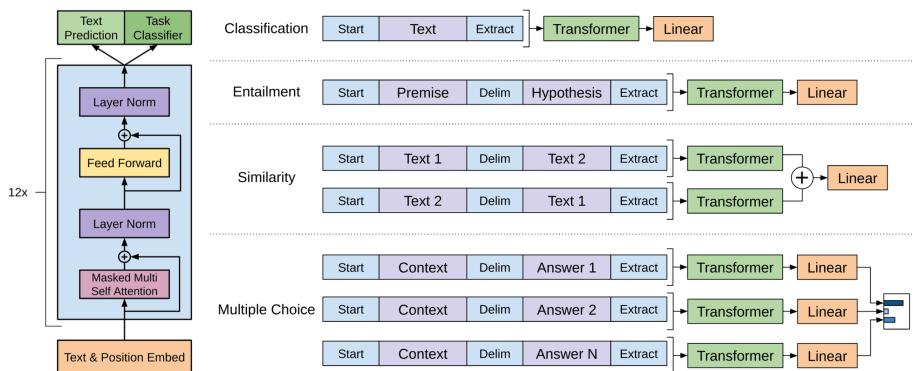


Рисунок взят из [оригинальной статьи GPT](#) .

Классификация отдельных предложений

Чтобы классифицировать отдельные предложения, просто подайте данные, как при обучении, и предскажите метку на основе окончательного представления последнего входного токена.

Примеры задач:

- SST-2 - бинарная классификация тональности (та, которую мы рассматривали на лекции «[Классификация текстов](#)»);
- CoLA (Корпус лингвистической приемлемости) — определяет, является ли предложение лингвистически приемлемым.

Классификация пар предложений



**Передача обучения**

Что такое трансферное обучение?

Обзор: Встраивание слов

Предварительно обученные модели

(Немного) адаптеров

(Заметка о) контрольных показателях

Анализ и интерпретируемость



Исследовательское мышление



Связанные статьи



Веселиться!



Чтобы классифицировать пары предложений, добавьте к двум фрагментам специальный токен-разделитель (например, `delim`). Затем предскажите метку на основе окончательного представления последнего входного токена.

Примеры задач:

- SNLI — классификация выводов. Для пары предложений определите, является ли второе выводом, противоречием или нейтральным;
- QQP (Quora Question Pairs) — по двум вопросам определите, эквивалентны ли они семантически;
- STS-B — для двух предложений определяется степень сходства от 1 до 5.

Ответы на вопросы и рассуждения с позиций здравого смысла

В этих задачах нам дается контекстный документ z , вопрос q , и набор возможных ответов $\{a_k\}$. Объедините документ и вопрос, а после токена-разделителя добавьте возможный ответ. Для каждого из возможных ответов обработайте соответствующие последовательности независимо с помощью модели GPT; затем нормализуйте с помощью слоя SoftMax для получения выходного распределения возможных ответов.

Примеры задач:

- **PAC** — понимание прочитанного. Дан отрывок, вопрос и несколько вариантов ответа. Выберите правильный.
- **Story Cloze** — понимание истории и изучение сценария. Дан рассказ из четырёх предложений и два возможных финала. Выберите правильный финал.

Что осталось : GPT-1-2-3, много шумихи и немного единорогов.

На данный момент существует три модели GPT:

- GPT-1 : Улучшение понимания языка с помощью генеративного предварительного обучения
- GPT-2 : языковые модели — это неконтролируемые многозадачные обучающиеся
- GPT-3 : Языковые модели учатся с небольшими усилиями

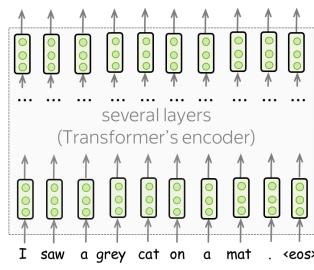
Эти модели различаются в основном объемом обучающих данных и числом параметров (см., например, [эту запись в блоге](#)). Обратите внимание, что эти модели настолько велики, что позволить себе обучать их могут только крупные компании. Это вызвало множество обсуждений проблем, связанных с использованием таких огромных моделей (этических, экологических и т. д.). Если вас это интересует, вы легко найдете массу информации в Интернете.

Но что вам непременно нужно увидеть, так это [сгенерированную GPT-2 историю о единорогах](#) в записи блога Open AI.

BERT : Двунаправленные кодеры, представленные трансформаторами

BERT был представлен в статье «BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding». Если ELMo получил награду за лучшую работу на конференции NAACL 2018, то BERT получил то же самое годом позже, на конференции NAACL 2019 :) Давайте попробуем понять, почему.

Архитектура модели BERT очень проста, и вы уже знаете, как она работает: это всего лишь кодер Трансформера. Новшеством являются цели обучения и способ использования BERT для последующих задач.



Model architecture:

- Transformer's encoder

What is special about it:

- Training objectives
 - MLM: Masked language modeling
 - NSP: Next sentence prediction
- The way it is used
 - No task-specific models

Как обучить (двунаправленный) кодер, используя простые тексты? Нам известна только цель моделирования языка слева направо, но она применима только для декодеров, где



Передача обучения

Что такое трансферное обучение?

Обзор: Встраивание слов

Предварительно обученные модели

(Немного) адаптеров

(Заметка о) контрольных показателях

Анализ и интерпретируемость



Исследовательское мышление



Связанные статьи



Веселиться!



каждый токен может использовать только предыдущие (и не видит будущее). Авторы BERT предложили другие цели обучения для немаркированных данных. Прежде чем перейти к ним, давайте сначала рассмотрим, что BERT подаёт на вход кодеру Transformer.

Входные данные для обучения : пары предложений со специальными токенами

В процессе обучения BERT видит пары предложений, разделённые специальным токеном-разделителем [SEP]. Другой специальный токен — [CLS]. В процессе обучения он используется для достижения цели NSP, которую мы рассмотрим далее. После обучения модель используется для последующих задач.

[CLS]: Special token

- Training time: predict if sentences are consecutive or not (Next Sentence Prediction /NSP objective)
- Test time: downstream tasks (e.g., classification)

[SEP]: Special token-separator

[CLS] My dog is very cute [SEP] He likes playing in the garden with me [SEP]

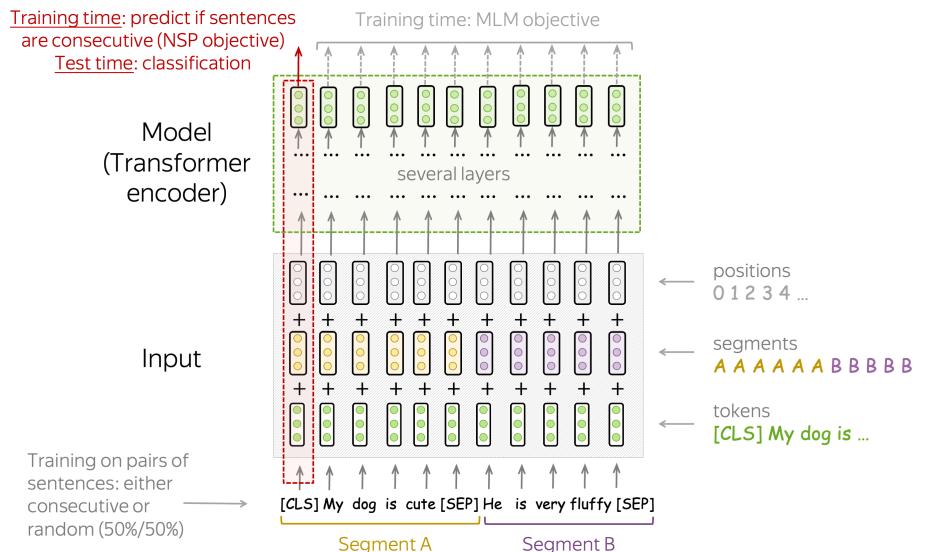
Segment A

Segment B

Training on pairs of sentences: either consecutive or random (50%/50%)

Чтобы модель могла легко различать эти предложения, помимо токеновых и позиционных вложений она использует сегментные вложения. В целом, входные данные для модели представляют собой сумму токеновых, позиционных и сегментных вложений. Эти представления подаются в преобразователь-кодировщик, и представления, расположенные выше, используются сначала для обучения, а затем для последующих приложений.

Лена : Хотя для приложений ниже по течению мы также можем использовать представления из середины модели. Мы узнаем об этом позже.



Цели предварительной подготовки : Цель прогнозирования следующего предложения (NSP)

Задача прогнозирования следующего предложения (NSP) — это задача бинарной классификации. Используя представление специального токена [CLS] на последнем слое, модель предсказывает, являются ли два предложения последовательными предложениями в некотором тексте или нет. Обратите внимание, что при обучении 50% примеров содержат последовательные предложения, извлеченные из обучающих текстов, а еще 50% — случайную пару предложений. Взгляните на пару примеров из [оригинальной статьи](#).

Ввод : [CLS] мужчина пошел в магазин [MASK] [SEP] он купил галлон [MASK] молока [SEP]
Метка : isNext

Ввод : [CLS] мужчина пошел в магазин [MASK] [SEP] пингвин [MASK] — это полет ## меньше птиц [SEP]

Метка : notNext**Передача обучения**

Что такое трансферное обучение?

Обзор: Встраивание слов

Предварительно обученные модели

(Немного) адаптеров

(Заметка о) контрольных показателях

Анализ и интерпретируемость



Исследовательское мышление



Связанные статьи



Веселиться!



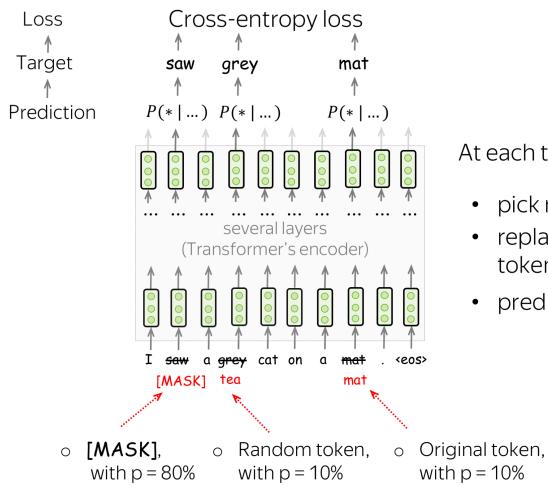
Это задание обучает модель понимать взаимосвязи между предложениями. Как мы увидим позже, это позволит использовать BERT для решения сложных задач, требующих рассуждений.

Цели предварительного обучения : Моделирование маскированного языка (MLM)

У BERT есть две цели обучения, и самая важная из них — это цель моделирования маскированного языка (MLM). При достижении цели MLM на шаге происходит следующее:

- выбрать несколько токенов
(каждый токен выбирается с вероятностью 15%)
- заменить эти выбранные токены
(на специальный токен [MASK] - с p=80%, на случайный токен - с p=10%, на исходный токен (остаться неизменным) - с p=10%)
- предсказать исходные токены (вычислить потери).

На иллюстрации ниже показан пример этапа обучения для одного предложения. Вы можете просмотреть слайды, чтобы увидеть весь процесс.



At each training step:

- pick randomly 15% of tokens
- replace each of the chosen tokens with something
- predict original chosen tokens



- 1.
- 2.
- 3.
- 4.
- 5.
- 6.
- 7.

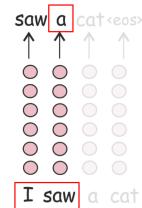
MLM по-прежнему является языковым моделированием: цель — предсказать некоторые лексемы в предложении/тексте на основе какой-либо его части. Чтобы прояснить ситуацию, сравним MLM со стандартным языковым моделированием слева направо.

На каждом шаге стандартные LM, работающие слева направо, предсказывают следующий токен на основе предыдущих. Это означает, что конечные представления, те из последнего слоя, которые используются для прогнозирования, кодируют только предыдущий контекст, то есть они не видят будущего.

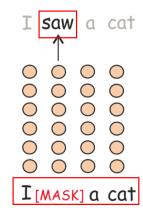
В отличие от этого, многонаправленные линейные модели (MLM) видят весь текст сразу, но некоторые токены искаются: именно поэтому BERT двунаправленный. Обратите внимание: чтобы ELMo распознавал как левый, так и

Language Modeling

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

Masked Language Modeling

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



left-to-right, does not see future

sees the whole text, but something is corrupted



Передача обучения

Что такое трансферное обучение?

Обзор: Встраивание слов

Предварительно обученные модели

(Немного) адаптеров

(Заметка о) контрольных показателях

Анализ и интерпретируемость



Исследовательское мышление



Связанные статьи



Веселиться!



правый контекст, авторам пришлось обучить два разных односторонних LM, а затем объединить их представления. В BERT это не нужно: достаточно одной модели.

Тонкая настройка: использование BERT для нисходящих задач

Теперь давайте рассмотрим, как применять BERT для различных задач. Пока что мы рассмотрим только простой вариант: когда предобученная модель точно настраивается для каждой из последующих задач. Позже мы рассмотрим другие способы адаптации модели для различных приложений (например, в разделе «[Адаптеры](#)»).

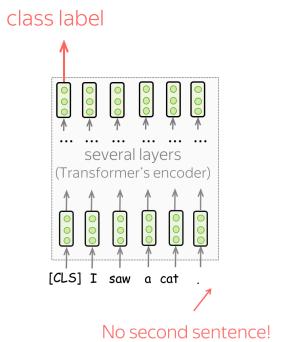
В этой части я упомяну лишь некоторые из задач. Подробнее о популярных наборах данных для оценки можно узнать на [сайте бенчмарка GLUE](#).

Классификация отдельных предложений

Чтобы классифицировать отдельные предложения, введите данные, как показано на иллюстрации, и спрогнозируйте метку на основе окончательного представления токена [CLS]

Примеры задач:

- SST-2 — бинарная классификация тональности (та, которую [мы рассматривали на лекции «Классификация текстов»](#));
- CoLA (Корпус лингвистической приемлемости) — определяет, является ли предложение лингвистически приемлемым.

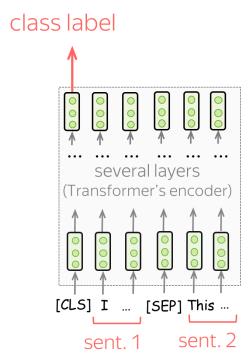


Классификация пар предложений

Для классификации пар предложений используйте данные, как при обучении. Аналогично классификации отдельных предложений, предскажите метку на основе финального представления токена [CLS].

Примеры задач:

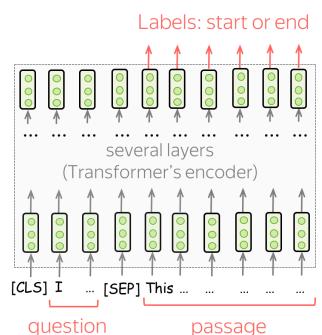
- SNLI — классификация выводов. Для пары предложений определите, является ли второе выводом, противоречием или нейтральным;
- QQP (Quora Question Pairs) — по двум вопросам определите, эквивалентны ли они семантически;
- STS-B — для двух предложений вывести оценку схожести от 1 до 5.



Вопрос Ответ

Для проверки качества авторы BERT использовали только один набор данных, SQuAD v1.1. В этом задании вам даётся текстовый фрагмент и вопрос. Ответ на этот вопрос всегда является частью текста, и задача состоит в том, чтобы найти правильный фрагмент текста.

Чтобы использовать BERT для решения этой задачи, введите вопрос и текст, как показано на рисунке. Затем для каждого токена в тексте используйте финальные представления BERT, чтобы предсказать, является ли этот токен началом или концом правильного сегмента.



Тегирование отдельных предложений

В задачах тегирования необходимо предсказать теги для каждого токена. Например, в задаче распознавания именованных сущностей (NER) необходимо предсказать, является ли слово



Передача обучения

Что такое трансферное обучение?

Обзор: Встраивание слов

Предварительно обученные модели

(Немного) адаптеров

(Заметка о) контрольных показателях



Анализ и интерпретируемость

Исследовательское мышление



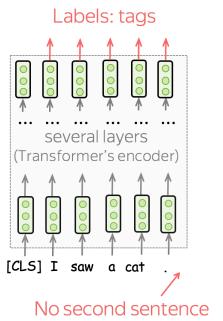
Связанные статьи



Веселиться!



именованной сущностью и её тип (например, местоположение, человек и т. д.).



(Немного) адаптеров: передача параметров с эффективностью

До сих пор мы рассматривали только стандартный способ передачи знаний от предобученных моделей (например, BERT) к последующим задачам: тонкая настройка. «Тонкая настройка» означает, что вы берёте предобученную модель и обучаете её для интересующей вас задачи (например, классификации тональности) с довольно низкой скоростью обучения. Это означает, что, во-первых, вы обновляете всю (большую) модель, а во-вторых, для каждой задачи вам необходимо настраивать отдельную копию предобученной модели. В итоге для нескольких последующих задач вы получаете множество больших моделей — это крайне неэффективно!

Finetuning:

- need to update the whole (huge!) model for each task

Parameters updated: 100% - **inefficient**

Adapters:

- model is fixed, train only small adapters

Parameters updated: e.g., ≈1% - **efficient**

В качестве альтернативы, в статье ICML 2019 года « [Parameter-Efficient Transfer Learning for NLP](#) » был предложен перенос с помощью адаптерных модулей. В этом случае параметры исходной модели фиксированы, и для каждой задачи требуется обучение лишь нескольких обучаемых параметров: эти новые параметры, специфичные для каждой задачи, называются адаптерами. Благодаря адаптерным модулям перенос становится очень эффективным: большая часть, предобученная модель, используется совместно всеми последующими задачами.

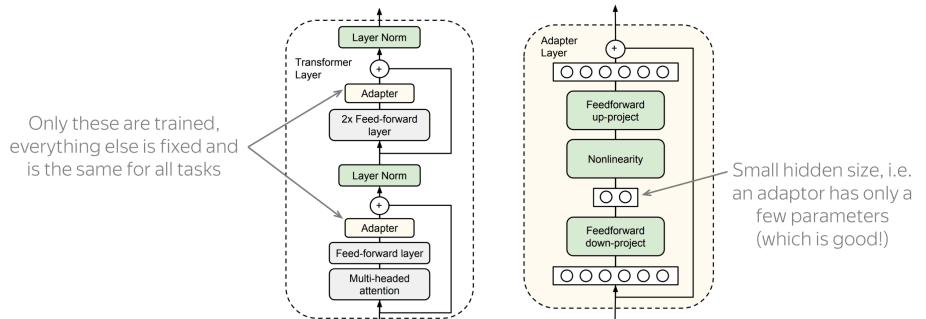


Рисунок взят из статьи «[Эффективное по параметрам переносное обучение для обработки естественного языка](#)».

Пример модуля адаптера и слоя преобразователя с адаптерами показан на рисунке. Как видите, модуль адаптера очень прост: это всего лишь двухслойная сеть прямого распространения с нелинейностью. Важно отметить, что скрытая размерность этой сети мала, а значит, и общее число параметров в адаптере также невелико. Именно это делает адаптеры очень эффективными.

Другие адаптеры и некоторые ресурсы

Сейчас существует множество различных модификаций адаптеров для множества задач и моделей. Например, репозиторий [AdapterHub](#) содержит предварительно обученные модули адаптеров. Поскольку это была системная демонстрация на конференции EMNLP 2020, она может стать хорошей отправной точкой для поиска ссылок на последние версии адаптеров ([Лена](#) : По крайней мере, на момент написания этой статьи: начало декабря 2020 года, то есть всего через пару недель после конференции EMNLP 2020).

(Заметка о) контрольных показателях

Курс НЛП | Для вас



Передача обучения

Что такое трансферное обучение?

Обзор: Встраивание слов

Предварительно обученные модели

(Немного) адаптеров

(Заметка о) контрольных показателях

Анализ и интерпретируемость



Исследовательское мышление



Связанные статьи



Веселиться!



Анализ и интерпретируемость

Сначала давайте вкратце рассмотрим методы анализа, которые мы уже использовали на предыдущих лекциях: изучение компонентов модели (например, нейронов в сверточных/длинноволновых сетях и головок внимания в Transformer), исследование лингвистической структуры (например, кодируют ли представления NMT информацию о морфологии) и оценка конкретных явлений на основе прогнозов модели (например, оценка согласования подлежащего и сказуемого в языковых моделях). Сегодня мы посмотрим, что произойдёт, если применить те же методы к BERT.

The methods we used previously:

- (model-specific) looking at model components
- (model-agnostic) probing for linguistic structure
- (model-agnostic) looking at predictions and evaluating specific phenomena

What we will see in this lecture:

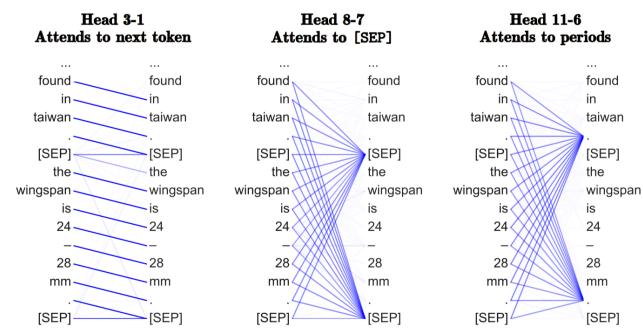
- Heads in Multi-Head Attention (BERT)
- FFNs as Key-Value Memories
- BERT and the classical NLP pipeline
- BERT as knowledge base

Компоненты модели: Головки внимания BERT

На предыдущих лекциях мы рассматривали фильтры CNN в Text Classification, нейроны LSTM в Language Models и головки внимания в NMT Transformer. Теперь давайте посмотрим, что исследователи обнаружили в паттернах внимания головок BERT!

Простые паттерны: позиционные головы, внимание к [SEP] и период

Помните позиционные головы с собственным вниманием, которые мы видели в Transformer для машинного перевода? Оказывается, у BERT тоже есть такие головки. В статье «[На что смотрит BERT? Анализ внимания BERT](#)» показано, что некоторые головки BERT имеют простые закономерности: не только позиционные (когда все токены обращают внимание на предыдущий или следующий токен), но и головки, направляющие всё своё внимание на некоторые токены, например, на особый токен [SEP] точки.



Примеры взяты из статьи «[На что обращает внимание BERT? Анализ внимания BERT](#)».

Синтаксические заголовки

Подобно синтаксическим головкам внутреннего внимания, обнаруженным в NMT Transformer, также было обнаружено несколько головок BERT, специализирующихся на отслеживании определённых

Передача обучения

Что такое трансферное обучение?

Обзор: Встраивание слов

Предварительно обученные модели

(Немного) адаптеров

(Заметка о) контрольных показателях

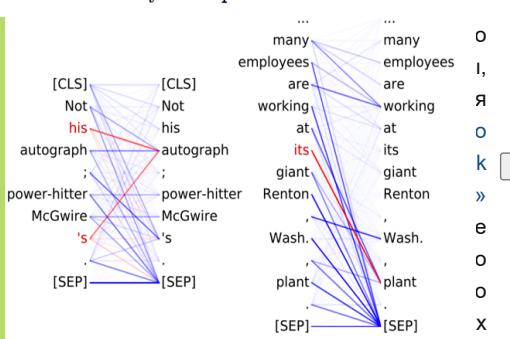
Анализ и интерпретируемость

Исследовательское мышление

Связанные статьи

Веселиться!

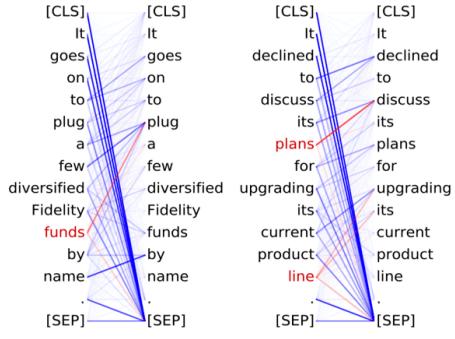
- Possessive pronouns and apostrophes attend to the head of the corresponding NP
- 80.5% accuracy at the poss relation



Примеры синтаксических головок из статьи «На что обращает внимание BERT? Анализ внимания BERT».

это делали. Как всегда, нужно быть очень внимательным :)

- Direct objects attend to their verbs
- 86.8% accuracy at the dobj relation



Примеры синтаксических головок из статьи «На что обращает внимание BERT? Анализ внимания BERT».

Компоненты модели: FFN как ключевая память

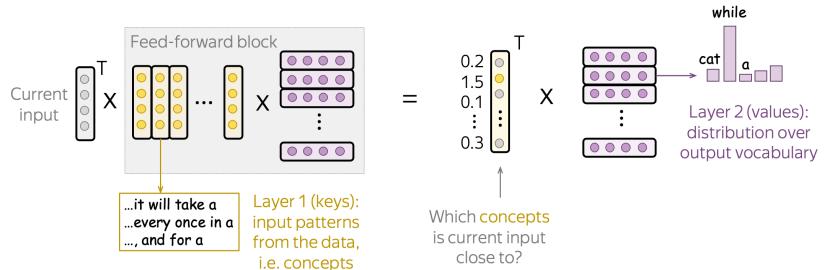
Трансформатор и остаточный поток

Как мы узнали из [предыдущей лекции](#), в преобразователе (кодере или декодере) представление токена развивается от встраивания входного токена до предсказания на конечном слое. Через остаточные связи блоки внимания и прямой связи обновляют исходное представление, добавляя новую информацию. Эта последовательность эволюционирующих представлений для одного и того же токена называется остаточным потоком.

Обратите внимание, что, хотя слои внимания позволяют обмениваться информацией между токенами, сети FFN работают в рамках того же остаточного потока. Теперь давайте рассмотрим этот слой подробнее.

Взгляд на FFN с точки зрения ключевого значения

В [данной статье](#) авторы предлагают рассматривать слои прямой связи в языковых моделях на основе преобразователя как память типа «ключ-значение». В этом представлении столбцы первого слоя FFN кодируют текстовые понятия, а строки второго слоя FFN кодируют распределения по словарному запасу.



Когда наш входной вектор попадает в блок FFN, он сначала сопоставляется с некоторыми концепциями, закодированными в первом слое, и получает веса (на нашей иллюстрации это показано жёлтым цветом). Затем эти веса активируют соответствующие строки второго слоя FFN. Эти строки (умноженные на веса) обновляют распределение выходных токенов, закодированное в остаточном потоке.

Исследование: BERT заново открывает классический конвейер НЛП

Курс НЛП | Для вас



Передача обучения

Что такое трансферное обучение?

Обзор: Встраивание слов

Предварительно обученные модели

(Немного) адаптеров

(Заметка о) контрольных показателях

Анализ и интерпретируемость



Исследовательское мышление



Связанные статьи



Веселиться!



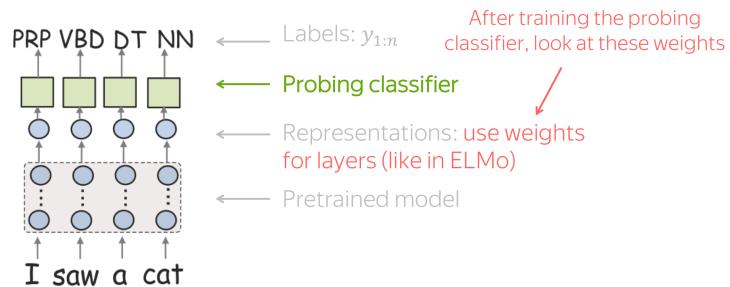
Сейчас существует множество статей, посвященных зондированию в BERT. В этой части давайте рассмотрим краткую статью ACL 2020 «BERT переосмысливает классический конвейер обработки естественного языка».

Как: зондирование с долей креативности

На предыдущей лекции мы изучили стандартное исследование лингвистической структуры:

- отправьте данные в вашу предварительно обученную модель,
- получить векторные представления из некоторого слова,
- обучить зондирующий классификатор предсказывать лингвистические метки на основе представлений,
- используйте его точность как меру того, насколько хорошо представления кодируют эти метки.

Конечно, это можно сделать и для BERT. Но у BERT много слоёв (например, 18 или 24), и чтобы понять, какие слои лучше кодируют определённую задачу, нам пришлось бы обучить 18 (или 24) пробных классификаторов для каждой задачи — это огромный объём работы!



К счастью, авторы нашли способ оценить все слои одновременно. Вместо того, чтобы выбирать представления из каждого слоя отдельно, для обучения зондирующего классификатора они взвешивают представления из всех слоёв и изучают веса с помощью зондирующего классификатора (прямо как в ELMo!). После обучения зондирующего классификатора веса могут служить мерой важности слоя для определённой задачи.

Что: Задачи зондирования границ

Авторы экспериментируют с набором [заданий на проверку границ](#). Это тестовые наборы для нескольких лингвистических задач. Ниже приведены несколько примеров.

- часть речи Я хочу найти больше, [что-то] больше или глубже. → NN (Существительное)
- составляющие Я хочу найти больше, [что-то большее или более глубокое] . → NP (Имущественная группа)
- зависимости [I]₁ я не [уверен]₂ Насколько же он надежен. → nsubj (именное подлежащее)
- объекты Самым захватывающим является лабиринт, известный как [Пещера Ветра] . → LOC (местоположение)
- маркировка семантической роли Я хочу [найти] ₁[что-то большее или более глубокое]₂ . → Arg1 (Агент)
- кореферентность Итак [последователи] ₁ хотел что-нибудь сказать о том, что [оны]₂ увидел . → Правда

Результаты: BERT следует классическому конвейеру НЛП



Передача обучения

Что такое трансферное обучение?

Обзор: Встраивание слов

Предварительно обученные модели

(Немного) адаптеров

(Заметка о) контрольных показателях

Анализ и интерпретируемость



Исследовательское мышление



Связанные статьи



Веселиться!



Результаты показаны на рисунке справа. Слой показаны слева направо, веса каждого слоя обозначены тёмно-синим цветом.

Мы видим, что по мере продвижения от нижнего к верхнему слою влияние каждого слоя сначала увеличивается, а затем уменьшается. Но главный результат заключается не в самих значениях, а в том, как BERT представляет различные лингвистические

задачи. В классическом NLP существовал порядок различных задач (именно в этом порядке, показанном на рисунке, и в том порядке, в котором я показал вам примеры для каждой задачи). Чтобы решить последующую задачу в классическом NLP, нужно было решить все предыдущие. Что интересно, BERT представляет эти задачи в том же порядке! Например, зависимости представлены позже, чем теги частей речи, а кореферентность выучивается позже, чем обе эти задачи.

In classical NLP, to solve a subsequent task is required to solve the previous one

BERT "solves" these tasks in the same order!

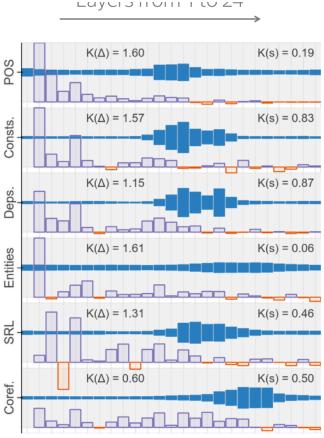
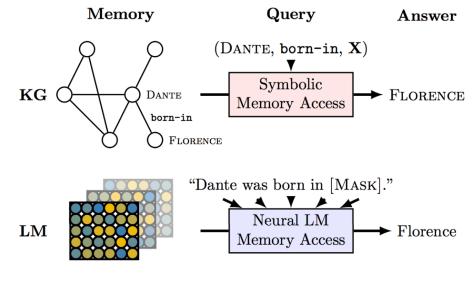


Рисунок с результатами взят из статьи «BERT заново открывает классический конвейер НЛП».

Взгляд на прогнозы: знают ли ELMo и BERT факты?

На лекции по языковому моделированию мы узнали, как оценивать языковые модели для конкретных явлений, анализируя их предсказания. В той лекции мы рассмотрели очень популярную задачу на согласование подлежащего и сказуемого. Сегодня мы займемся чем-то гораздо более интересным — проверим, знает ли модель факты! В качестве примера мы используем статью EMNLP 2019 года «Языковые модели как базы знаний?».

Обычно фактические знания хранятся в базах знаний, содержащих триады (субъект, отношение, объект), например, (Данте, уроженец, Флоренция), как показано на рисунке. Однако такие базы знаний сложно получить: для извлечения знаний обычно приходится использовать сложные конвейеры.



e.g. ELMo/BERT

Рисунок с результатами взят из статьи « Языковые модели как базы знаний?»

Но что, если предобученные языковые модели (ELMo, BERT) уже знают факты? Давайте проверим! Вместо триплетов отношений мы передадим модели предложение в стиле «клоуз» с замаскированным объектом. Например, мы даём нашей модели предложение «Данте родился в ____» и просим её предсказать токен вместо маски. Обратите внимание, что мы делаем это без какой-либо тонкой настройки — просто модель языка!

Оказывается, предобученные модели могут довольно хорошо знать факты. Обратите внимание, что под «знать» мы здесь не подразумеваем, что они что-то понимают, а лишь то, что статистика обучающих данных, собранная в этих моделях, может быть использована для извлечения фактов. Взгляните на несколько примеров ниже (в оригинальной статье их больше!).

Query	Answer	Generation (model log-probability)
Francesco Bartolomeo Conti was born in ____.	Florence	Rome [-1.8], Florence [-1.8], Naples [-1.9], Milan [-2.4], Bologna [-2.5]
Adolphe Adam died in ____.	Paris	Paris [-0.5], London [-3.5], Vienna [-3.6], Berlin [-3.8], Brussels [-4.0]
English bulldog is a subclass of ____.	dog	dogs [-0.3], breeds [-2.2], dog [-2.4], cattle [-4.3], sheep [-4.5]
The official language of Mauritius is ____.	English	English [-0.6], French [0.9], Arabic [-6.2], Tamil [-6.7], Malayalam [-7.0]
Patrick Oboya plays in ____ position.	midfielder	centre [-2.0], center [-2.2], midfielder [-2.4], forward [-2.4], midfield [-2.7]
Hamburg Airport is named after ____.	Hamburg	Hess [-7.0], Hermann [-7.1], Schmidt [-7.1], Hamburg [-7.5], Ludwig [-7.5]

Примеры взяты из статьи « Языковые модели как базы знаний?»

Что ещё интереснее, мы можем рассматривать не только формальные факты, но и общеизвестные факты: взгляните на примеры. Подробнее о построении тестового набора см. в статье.



Передача обучения

Что такое трансферное обучение?

Обзор: Встраивание слов

Предварительно обученные модели

(Немного) адаптеров

(Заметка о) контрольных показателях

Анализ и интерпретируемость



Исследовательское мышление



Связанные статьи



Веселиться!



Query	Answer	Generation (model log-probability)
You are likely to find a overflow in a _____. Ravens can _____. Joke would make you want to _____. Sometimes virus causes _____. Birds have _____. Typing requires _____. Time is _____. You would celebrate because you are _____. Skills can be _____. A pond is for _____.	drain fly laugh infection feathers speed finite alive taught fish	sewer [-3.1], canal [-3.2], toilet [-3.3], stream [-3.6], drain [-3.6] fly [-1.5], fight [-1.8], kill [-2.2], die [-3.2], hunt [-3.4] cry [-1.7], die [-1.7], laugh [-2.0], vomit [-2.6], scream [-2.6] disease [-1.2], cancer [-2.0], infection [-2.6], plague [-3.3], fever [-3.4] wings [-1.8], nests [-3.1], feathers [-3.2], died [-3.7], eggs [-3.9] patience [-3.5], precision [-3.6], registration [-3.8], accuracy [-4.0], speed [-4.1] short [-1.7], passing [-1.8], precious [-2.9], irrelevant [-3.2], gone [-4.0] happy [-2.4], human [-3.3], alive [-3.3], young [-3.6], free [-3.9] acquired [-2.5], useful [-2.5], learned [-2.8], combined [-3.9], varied [-3.9] swimming [-1.3], fishing [-1.4], bathing [-2.0], fish [-2.8], recreation [-3.1]

Примеры взяты из статьи « Языковые модели как базы знаний? »

Вернуться к ключевому значению FFN

Мы увидели, что BERT и другие, похоже, знают некоторые факты. Но как эти факты хранятся в модели? Оказывается, некоторые из них закодированы в FFN!

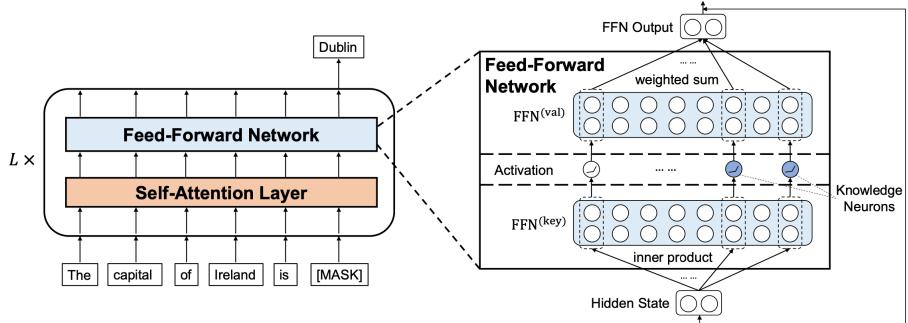


Рисунок взят из статьи « [Нейроны знаний в предварительно обученных трансформаторах](#) » .

Вернёмся к представлению о ключе-значении в памяти FFN. В статье « [Нейроны знаний в предобученных трансформаторах](#) » рассматривались триплеты (субъект, отношение) -> объект и было обнаружено, что некоторые нейроны внутри FFN кодируют эти отношения! В частности, каждый нейрон активируется всякий раз, когда на вход поступает пара (субъект, отношение), и запускает строку, кодирующую объект .



Исследовательское мышление

Как

- Прочитайте краткое описание в начале — это наша отправная точка, нечто известное.
- Прочитайте вопрос и подумайте: минуту, день, неделю... — дайте себе время! Даже если вы не думаете об этом постоянно, что-то всё равно может прийти в голову.
- Посмотрите на возможные ответы — предыдущие попытки решить эту задачу. Важно: не обязательно предлагать что-то в точности похожее — помните, каждая статья обычно занимает у авторов несколько месяцев. Важна привычка думать о таких вещах! Всё остальное, что нужно учёному, — это время: пробовать, ошибаться, думать, пока не получится.

Известно, что изучение чего-либо проходит легче, если не сразу получить ответ, а сначала подумать над ним. Даже если вы не хотите становиться исследователем, это всё равно хороший способ учиться!

Здесь будут упражнения!



Эта часть будет время от времени расширяться.



Связанные статьи

Передача обучения

Что такое трансферное обучение?

Обзор: Встраивание слов

Предварительно обученные модели

(Немного) адаптеров

(Заметка о) контрольных показателях

Анализ и интерпретируемость



Исследовательское мышление



Связанные статьи



Веселиться!



Как

- Высокий уровень : просмотрите основные результаты в кратких обзورах — получите представление о том, что происходит в данной области.
- Немного глубже : по темам, которые вас интересуют, читайте более подробные обзоры с иллюстрациями и пояснениями. Просмотрите ход рассуждений авторов и их ключевые наблюдения.
- Подробный анализ : прочтите понравившиеся статьи. Теперь, когда вы поняли основную идею, будет проще!

Здесь будут статьи!

Статьи будут появляться постепенно.



Веселиться!

Вскоре!

Мы все еще работаем над этим!



Последнее обновление: 12 июня 2025 г.