Порождение текста и машинный перевод.

Алексей Андреевич Сорокин

МГУ им. М. В. Ломоносова
весенний семестр 2022–2023 учебного года
Межфакультетский курс “Введение в компьютерную лингвистику” 29 марта, занятие 6

Нейронный машинный перевод

- Нейронный машинный перевод — задача условного порождения текста.
- Обычная вероятностная модель порождает текст на основе предыдущих слов.

$$w_i \sim p(w|h_{i-1})$$

h_{i-1} – состояние языковой модели после $i-1$ слова

Нейронный машинный перевод

- Нейронный машинный перевод — задача условного порождения текста.
- Обычная вероятностная модель порождает текст на основе предыдущих слов.

$$\begin{aligned} w_i &\sim p(w|h_{i-1}) \\ h_{i-1} &- \text{состояние языковой модели после } i-1 \text{ слова} \end{aligned}$$

- Условная вероятностная модель:

$$\begin{aligned} w_i &\sim p(w|h_{i-1}, c) \\ c &- \text{глобальный контекст} \end{aligned}$$

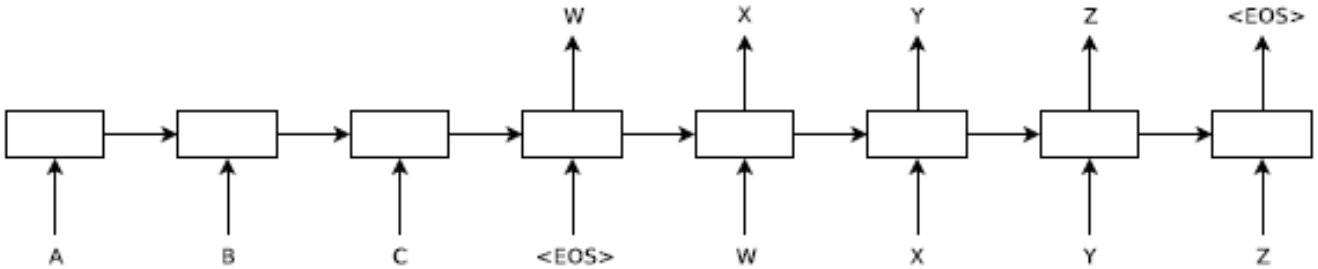
- Основная проблема: как вычислять c.

Нейронный машинный перевод

- Базовая модель: кодировщик-декодировщик (encoder-decoder):

$$\begin{aligned} c &= LSTM(x_1, \ldots, x_m), \\ x_1 \ldots x_m &- \text{исходное предложение} \end{aligned}$$

- Вектор c запоминает всю информацию об исходном предложении в одном векторе.



Нейронный машинный перевод

- Обычно на каждый шаг декодера явно подаётся предыдущее слово:

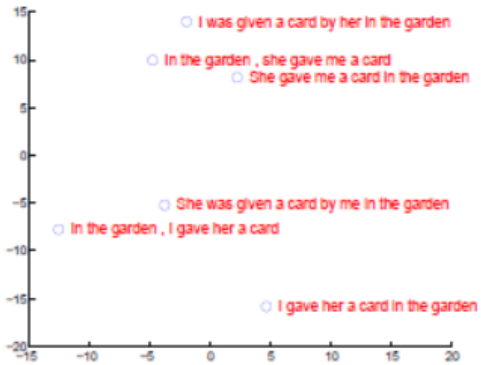
$$p(y_t | [\![y_1, \ldots, y_{t-1}]\!], c) = g(y_{t-1}, s_t, c)$$

- Как и энкодер, так и декодер включают в себя несколько слоёв.
- Входом следующего служит выход предыдущего.

Вектора предложений

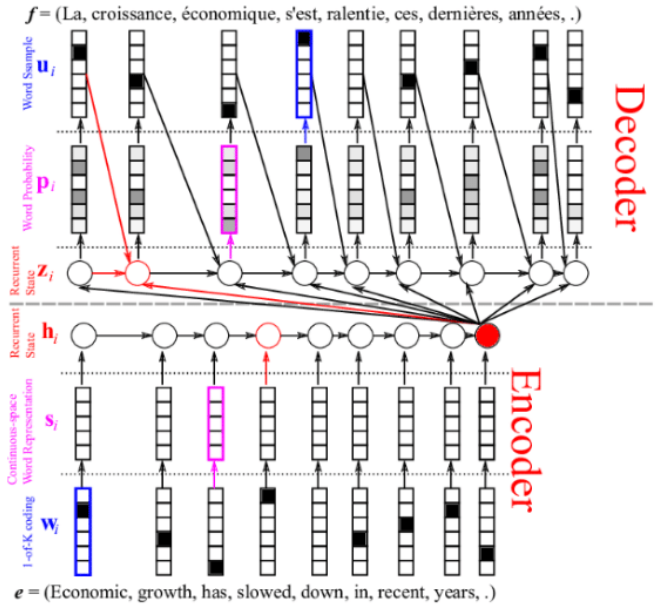
Секвенциальные модели

Вектора предложений



Нейронный машинный перевод

Нейронный машинный перевод



Обучение нейронных систем

- Многоклассовые нейронные классификаторы обучаются с помощью кросс-энтропии:

$$L_0(y, y') = -\sum y_j \log y'_j = -\log(y'_{\hat{j}})$$

- y_i, y'_i — правильное и предсказанное вероятностное распределения.

Обучение нейронных систем

- Многоклассовые нейронные классификаторы обучаются с помощью кросс-энтропии:

$$L_0(y, y') = -\sum y_j \log y'_j = -\log(y'_{\hat{j}})$$

- y_i, y'_i — правильное и предсказанное вероятностное распределения.
- Можно вычислять кросс-энтропию для каждого слова

$$\begin{aligned} L(\mathbf{t}, \mathbf{t}') &= -\sum_j L_0(t_i, t'_i) \\ L_0(t_i, t'_i) &= -\sum_j t_{ij} \log t'_{ij} \end{aligned}$$

- Здесь в каждой позиции предсказывается вероятностное распределение для текущего переводного слова.

Обучение нейронных систем

- Многоклассовые нейронные классификаторы обучаются с помощью кросс-энтропии:

$$L_0(y, y') = -\sum y_j \log y'_j = -\log (y'_{\hat{j}})$$

- y_i, y'_i — правильное и предсказанное вероятностное распределения.
- Можно вычислять кросс-энтропию для каждого слова

$$\begin{aligned} L(\mathbf{t}, \mathbf{t}') &= -\sum_j L_0(t_i, t'_i) \\ L_0(t_i, t'_i) &= -\sum_j t_{ij} \log t'_{ij} \end{aligned}$$

- Здесь в каждой позиции предсказывается вероятностное распределение для текущего переводного слова.
- Конец предложения: специальный символ END.
- Позиции после конца предложения: специальный символ PAD.

Обучение нейронных систем: функции штрафа и метрики

- Модели нейронного машинного перевода обучаются минимизировать кросс-энтропию.
- Эта функция штрафа не имеет отношения к реальному качеству перевода.

Обучение нейронных систем: функции штрафа и метрики

- Модели нейронного машинного перевода обучаются минимизировать кросс-энтропию.
- Эта функция штрафа не имеет отношения к реальному качеству перевода.
- Стандартная метрика для машинного перевода – BLEU.
- Основана на точности по словам и энграммам из правильного перевода (проверяет, что перевод не содержит лишних слов).
- Есть дополнительный штраф за слишком короткий ответ.

Обучение нейронных систем: функции штрафа и метрики

- Модели нейронного машинного перевода обучаются минимизировать кросс-энтропию.
- Эта функция штрафа не имеет отношения к реальному качеству перевода.
- Стандартная метрика для машинного перевода – BLEU.
- Основана на точности по словам и энграммам из правильного перевода (проверяет, что перевод не содержит лишних слов).
- Есть дополнительный штраф за слишком короткий ответ.
- BLEU неидеально коррелирует с оценкой качества человеком, но ничего лучше не придумано.
- BLEU недифференцируема, поэтому её нельзя оптимизировать градиентным спуском.

Обучение нейронных систем: некорректная позиция в предложении

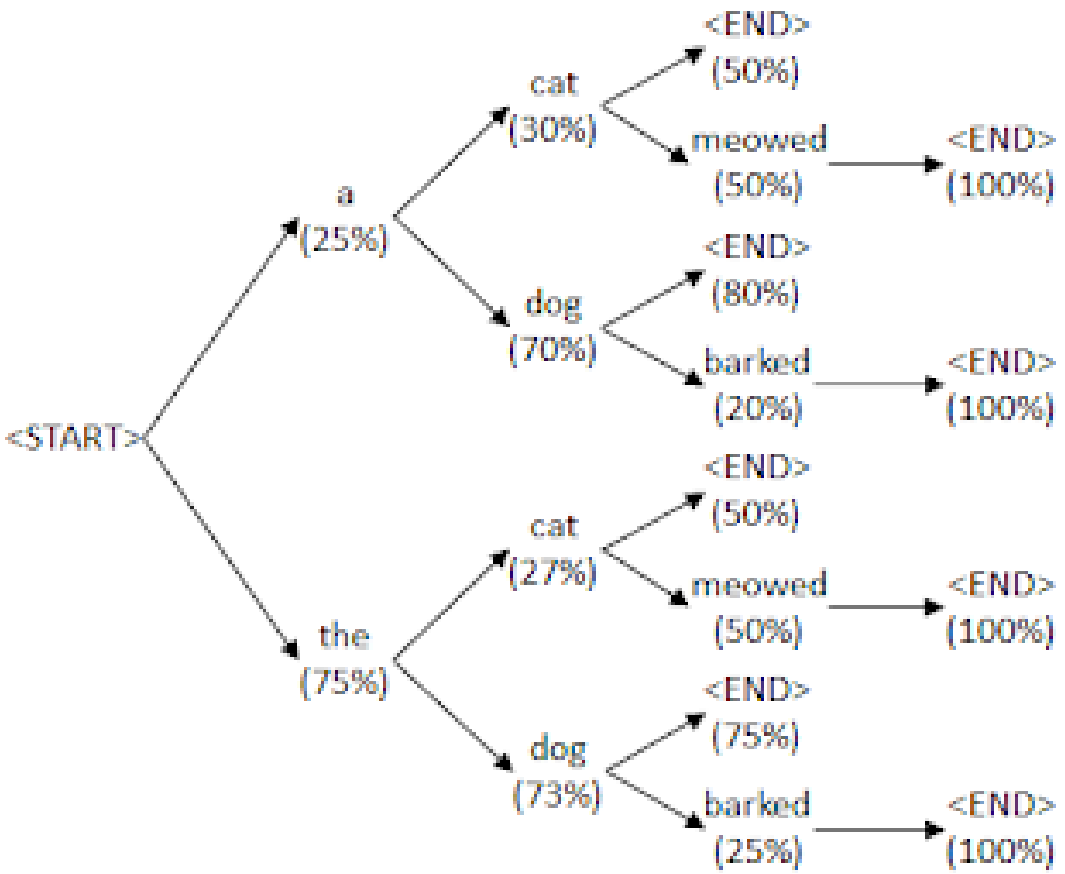
- Система может предсказать несколько слов вместо одного:

This is	the shortest	way home
Это	кратчайший	путь домой
Это	самый короткий	путь домой

- После “самый короткий” правильным словом будет считаться “домой”.

Машинный перевод: декодирование

- Стандартный способ декодирования в машинном переводе – поиск по лучу:



Машинный перевод: декодирование

- Стандартный способ декодирования в машинном переводе – поиск по лучу (beam search):
  - Поддерживается k наилучших гипотез (часто $k \in 5 \dots 10$).
  - На каждом шаге считаются все возможные продолжения существующих гипотез и пересчитываются их вероятности.
  - После этого снова выбираются k наилучших продолжений.
- Частный случай $k = 1$ – жадный поиск.

Машинный перевод: декодирование

- Стандартный способ декодирования в машинном переводе – поиск по лучу (beam search):
  - Поддерживается k наилучших гипотез (часто $k \in 5 \dots 10$).
  - На каждом шаге считаются все возможные продолжения существующих гипотез и пересчитываются их вероятности.
  - После этого снова выбираются k наилучших продолжений.
- Частный случай $k = 1$ – жадный поиск.
- Недостатки алгоритма:
  - Небольшое k – будет отсекаться много полезного.
  - Большое k – будут появляться вероятные ответы “общей тематики” (не связанные с конкретным входом).

Упрощение генерации

- Генерация последовательностей – сложная задача (и вычислительно, и алгоритмически).
- Иногда её можно упростить, сведя к классификации или разметке последовательности:

$$\text{корова} \mapsto \text{коровой}$$
$$\text{корова} \mapsto \text{а-ой}.$$

Упрощение генерации

- Генерация последовательностей – сложная задача (и вычислительно, и алгоритмически).
- Иногда её можно упростить, сведя к классификации или разметке последовательности:

$$\begin{aligned} \text{корова} &\mapsto \text{коровой} \\ \text{корова} &\mapsto \text{а-ой}. \end{aligned}$$

- Более сложные шаблоны тоже можно упростить:

$$\begin{aligned} \text{песок} &\mapsto \text{песком} && (1+o+2)\text{-}(1+2+ом) \\ \text{volver} &\mapsto \text{vuelvo} && (1+o+2+er)\text{-}(1+ue+2+o) \end{aligned}$$

Упрощение генерации

- Генерация последовательностей – сложная задача (и вычислительно, и алгоритмически).
- Иногда её можно упростить, сведя к классификации или разметке последовательности:

$$\begin{aligned} \text{корова} &\mapsto \text{коровой} \\ \text{корова} &\mapsto \text{а-ой}. \end{aligned}$$

- Более сложные шаблоны тоже можно упростить:

$$\begin{aligned} \text{песок} &\mapsto \text{песком} && \text{(1+о+2)-(1+2+ом)} \\ \text{volver} &\mapsto \text{vuelvo} && \text{(1+o+2+er)-(1+ue+2+o)} \end{aligned}$$

- Суммаризация (extractive summarization) в простейших случаях сводится к разметке последовательности (KEEP, DELETE, специальные операции).

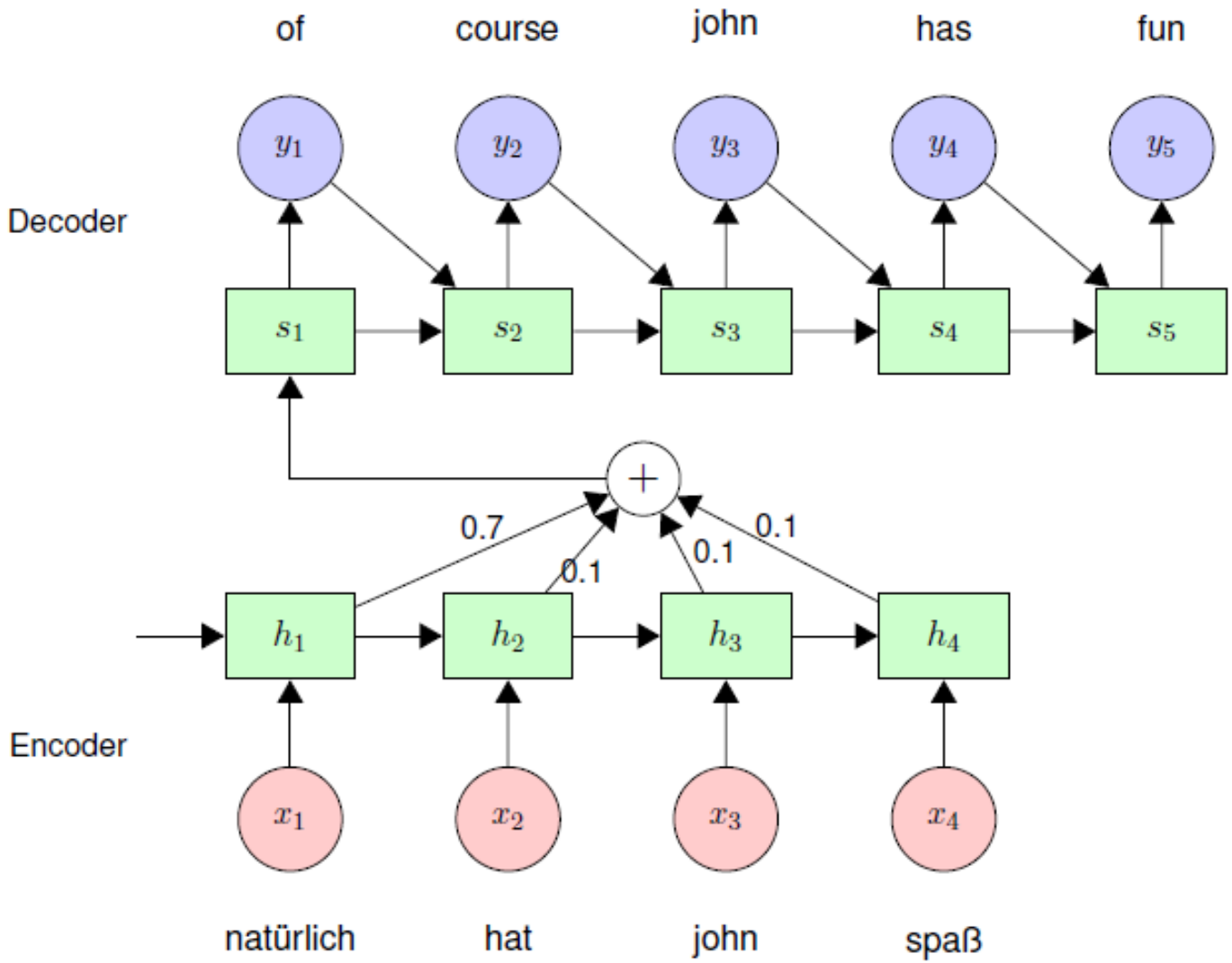
Механизм внимания: мотивация

- При сжатии всего предложения в один вектор часть информации неминуемо теряется.
- На разных этапах порождения выходного предложения полезна разная информация об исходных словах.

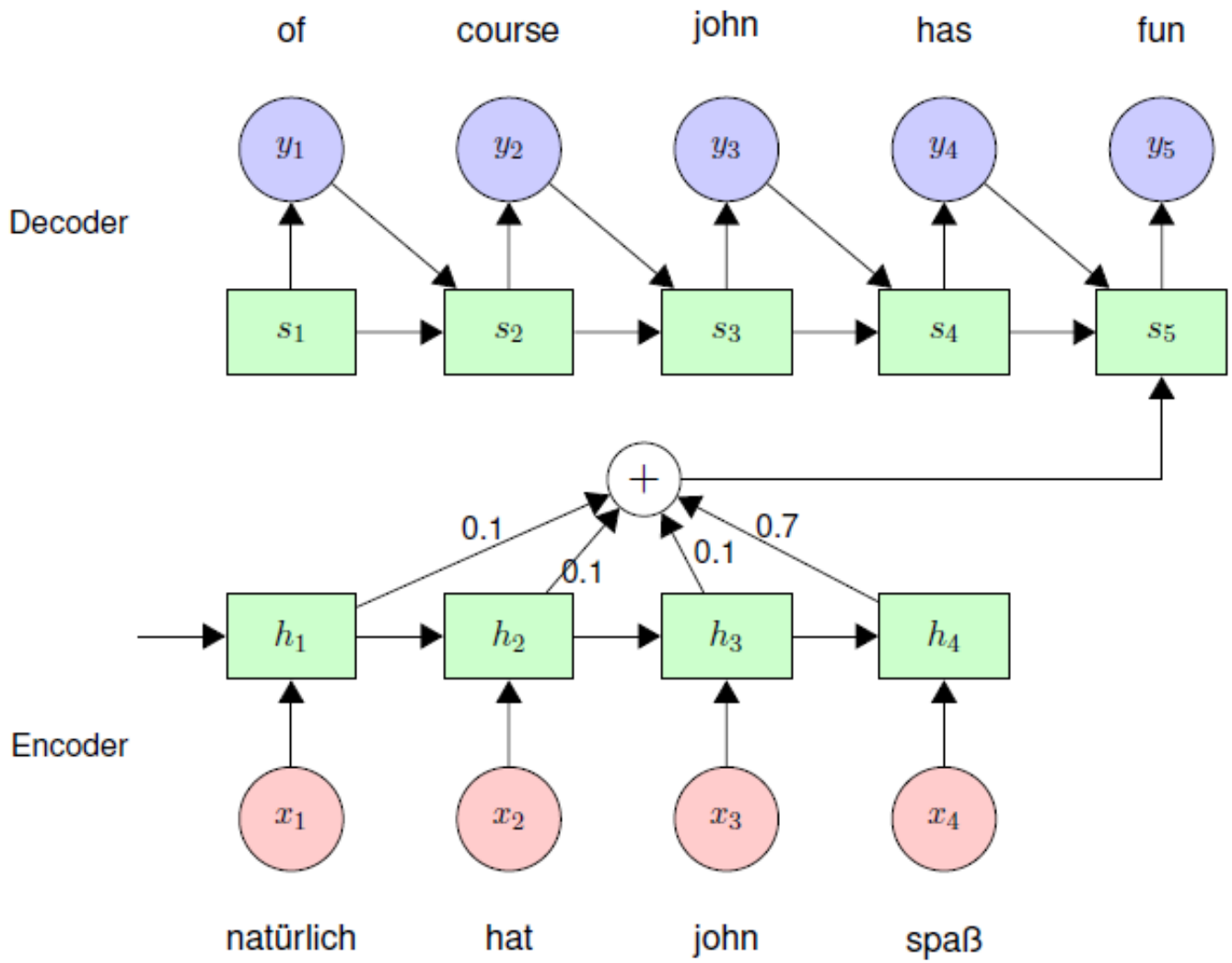
Механизм внимания: мотивация

- При сжатии всего предложения в один вектор часть информации неминуемо теряется.
- На разных этапах порождения выходного предложения полезна разная информация об исходных словах.
- Выход: сделать контекстный вектор зависящим от позиции в порождаемом предложении.

Механизм внимания: иллюстрация



Механизм внимания: иллюстрация



Механизм внимания: реализация

- Каждое следующее слово порождается отдельным вектором контекста:

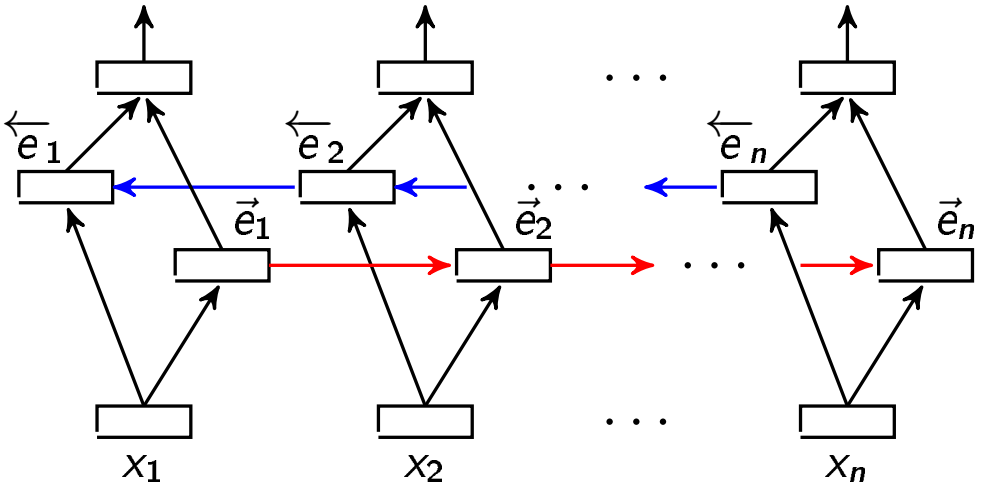
$$\begin{aligned} w_{i+1} &\sim p(h_i, c_i) \\ h_i &- \text{состояние декодера после } i \text{ слов}, \\ c_i &- \text{исходный контекст после } i \text{ слов}. \end{aligned}$$

- Вектор контекста – сумма векторов контекста для отдельных позиций:

$$\begin{aligned} c_i &= \sum_j \alpha_{ij} e_j, \\ e_j &- \text{вектор контекста в позиции } j, \\ \alpha_{ij} &- \text{мера влияния } e_j \text{ на } c_i. \end{aligned}$$

Механизм внимания: реализация

- Как считать α_{ij} и вектора e_j?
- e_j вычисляется с помощью двунаправленной рекуррентной сети (конкатенация $\vec{e}_j$ и $\overleftarrow{e}_j$):



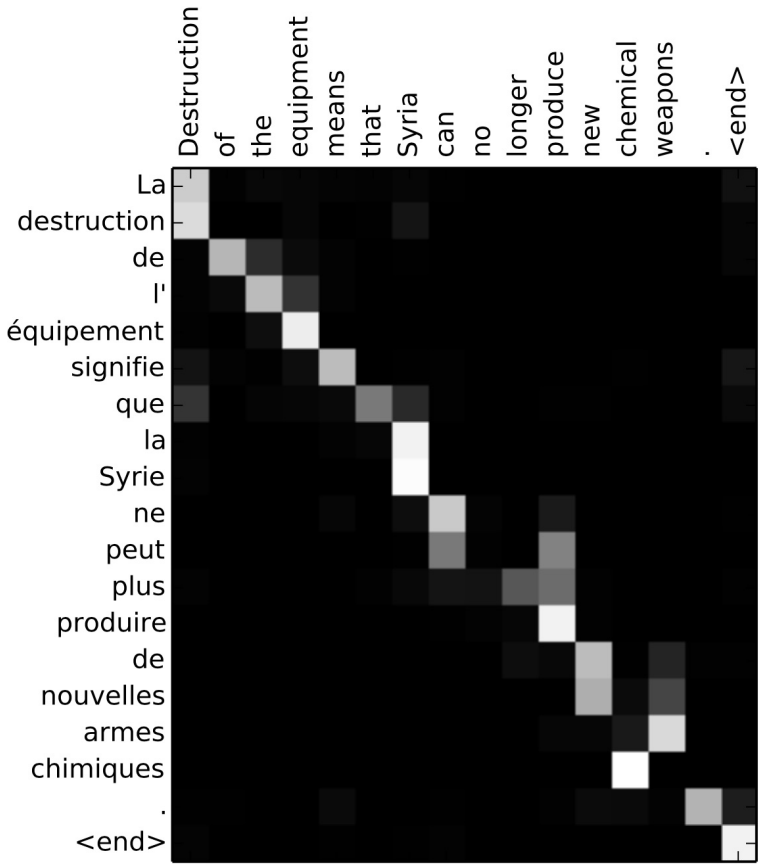
Механизм внимания: реализация

- α_{ij} можно считать разными способами.
- Bahdanau et al., 2015, Jointly learning to align and translate использовалось:

$$\begin{aligned} \alpha_{ij} &= \mathrm{softmax}(u_{ij}), \\ u_{ij} &= a(h_{i-1}, e_j), \\ h_{i-1} &- \text{состояние декодера на предыдущем шаге.} \end{aligned}$$

Механизм внимания: интерпретация

- Механизм внимания показывает, какие слова исходного текста влияют на слова сгенерированного текста.



Механизм внимания: вариации

- Механизм внимания показывает, какие слова исходного текста влияют на слова сгенерированного текста.
- Конкретные формулы могут отличаться.
- Задача формулы — вычислить числа u_{ij}, показывающие связь исходного вектора e_j с вектором контекста h_i в переводном тексте
- Далее эти формулы будут переведены в вероятности.

$$\alpha_{ij} = \mathrm{softmax}(u_{ij})$$

Механизм внимания: вариации

- Luong et al., Effective Approaches to Attention-based Neural Machine Translation: 3 формулы для механизма внимания.

$$
\begin{aligned}
u_{ij} &= \langle h_i, e_j \rangle && \text{(скалярное произведение),} \\
u_{ij} &= h_i^T W_a e_j && \text{(скалярное произведение} \\
& && \text{(с обученной матрицей),} \\
u_{ij} &= v_a^T \tanh(W_a[h_i, e_j]) && \text{(однослойная сеть)}
\end{aligned}
$$

- Лучше работает второй подход, но он требует больше ресурсов, чем первый.

Механизм внимания: вариации

- Luong et al., Effective Approaches to Attention-based Neural Machine Translation: 3 формулы для механизма внимания.

$$
\begin{aligned}
u_{ij} &= \langle h_i, e_j \rangle && \text{(скалярное произведение),} \\
u_{ij} &= h_i^T W_a e_j && \text{(скалярное произведение} \\
& && \text{(с обученной матрицей),} \\
u_{ij} &= v_a^T \tanh(W_a[h_i, e_j]) && \text{(однослойная сеть)}
\end{aligned}
$$

- Лучше работает второй подход, но он требует больше ресурсов, чем первый.
- Третий подход (Bahdanau et al., 2015) проигрывает первым двум.

Механизм внимания: вариации

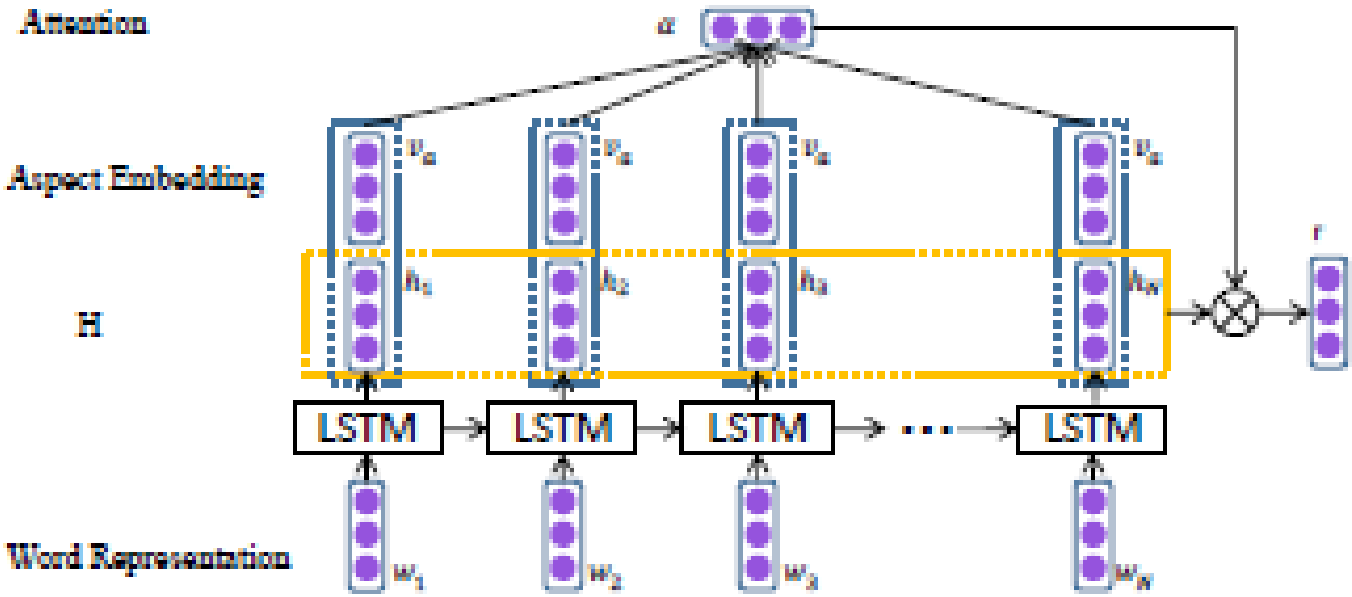
- Внимание можно использовать и для классификации.
- Самый простой способ получить вектор для предложения — взять взвешенную сумму векторов для его слов.
- Именно это делает tf-idf.

Механизм внимания: вариации

- Внимание можно использовать и для классификации.
- Самый простой способ получить вектор для предложения — взять взвешенную сумму векторов для его слов.
- Именно это делает tf-idf.
- Также применялись веса, зависящие от частей речи.
- Можно настроить веса под задачу:

$$
\begin{aligned}
s &= \sum \alpha_i x_i \qquad x_i \text{ — представление } i\text{-го слова,} \\
\alpha_i &= \frac{\exp(e_i)}{\sum\limits_j \exp(e_j)} \\
e_i &= \langle v_a, h_i \rangle, \\
h_i &= f(x_1, \ldots, x_n) \\
f &- \text{свёрточная или рекуррентная сеть для контекста}
\end{aligned}
$$

Простой механизм внимания: иллюстрация



[Wang et al., 2016. Attention-based LSTM for Aspect-level Sentiment Classification]

Простой механизм внимания для анализа тональности

- Механизм внимания работает с обучаемым представлением аспекта v_a (для ресторана стоимость, обслуживание и т.д.) и выдачей энкодера, не зависящей от аспекта.
- Это означает, что модель можно обучать одновременно на несколько аспектов.

Механизм внимания: переобозначение

- Текущая формула (переобозначения):

$$
\begin{aligned}
h &= \sum_i \alpha_i h_i^{value}, \\
\alpha_i &\sim \exp\left(\langle h_i^{key}, s\rangle\right), \\
s &- \text{глобальный вектор “запроса” (query)}, \\
h_i^{value} &- \text{“эмбеддинг-значение” (value)}, \\
h_i^{key} &- \text{“эмбеддинг-ключ” (key)}.
\end{aligned}
$$

- Откуда взять h_i^{value}, h_i^{key}.

Механизм внимания: переобозначение

- Текущая формула (переобозначения):

$$\begin{aligned} h &= \sum_i \alpha_i h_i^{value}, \\ \alpha_i &\sim \exp\left(\langle h_i^{key}, s\rangle\right), \\ s &- \text{глобальный вектор “запроса” (query)}, \\ h_i^{value} &- \text{“эмбеддинг-значение” (value)}, \\ h_i^{key} &- \text{“эмбеддинг-ключ” (key)}. \end{aligned}$$

- Откуда взять h_i^{value}, h_i^{key}.
- Проще всего вставить один слой персептрона:

$$\begin{aligned} h_i^{value} &= g(W^{value} h_i), \\ h_i^{key} &= g(W^{key} h_i) \end{aligned}$$

Механизм внимания: матричный вид

- Всё можно переписать в матричном виде:

$$\begin{aligned} h &= A_{1\times L} V_{L\times d}, \\ A &= \mathrm{softmax}(q_{1\times d} K^T_{L\times d}), \\ V &= g(H_{L\times d} W^{value}_{d\times d}), \\ K &= g(H_{L\times d} W^{key}_{d\times d}) \end{aligned}$$

- Финальная формула:

$$h = \mathrm{softmax}(QK^T)V$$

Механизм внимания: матричный вид

- Всё можно переписать в матричном виде:

$$\begin{aligned} h &= A_{1\times L} V_{L\times d}, \\ A &= \mathrm{softmax}(q_{1\times d} K^T_{L\times d}), \\ V &= g(H_{L\times d} W^{value}_{d\times d}), \\ K &= g(H_{L\times d} W^{key}_{d\times d}) \end{aligned}$$

- Финальная формула:

$$h = \mathrm{softmax}(QK^T)V$$

- На практике добавляют нормализующий множитель:

$$h = \mathrm{softmax}(\frac{QK^T}{\sqrt{d}})V$$

- Без него обучение более нестабильное.

Механизм самовнимания : матричный вид

- Механизм внимания используется, чтобы посчитать состояние для всего предложения с учётом всех слов.
- А что если так же считать новые состояния для всех слов?
- В этом случае даже удалённые слова будут влиять на текущий вектор (проблема для рекуррентных сетей).

Механизм самовнимания : матричный вид

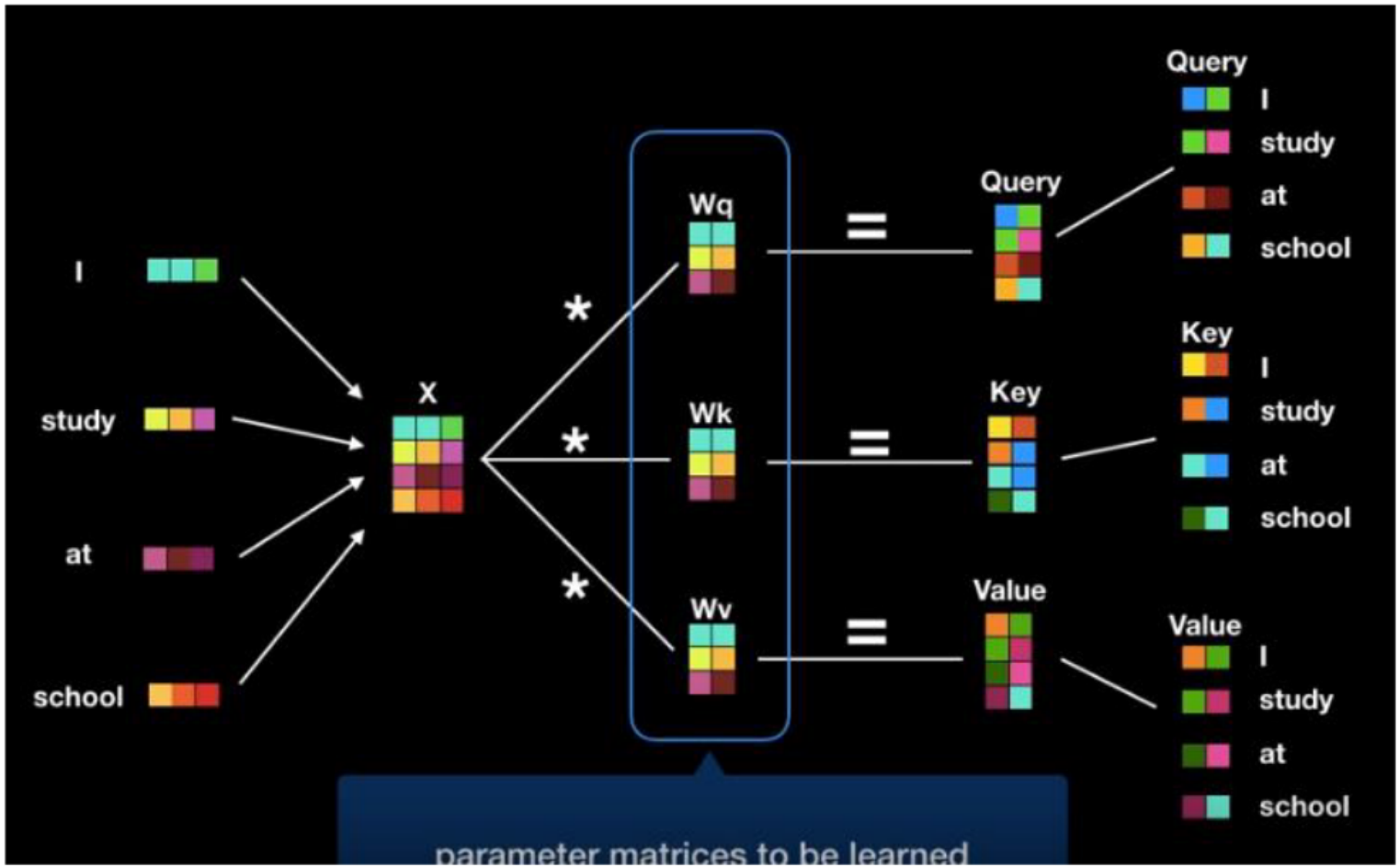
- Механизм внимания используется, чтобы посчитать состояние для всего предложения с учётом всех слов.
- А что если так же считать новые состояния для всех слов?
- В этом случае даже удалённые слова будут влиять на текущий вектор (проблема для рекуррентных сетей).
- Достаточно сделать "запрос" Q матрицей:

$$Q_{L\times d} = g(H_{L\times d} W^{query}_{d\times d})$$

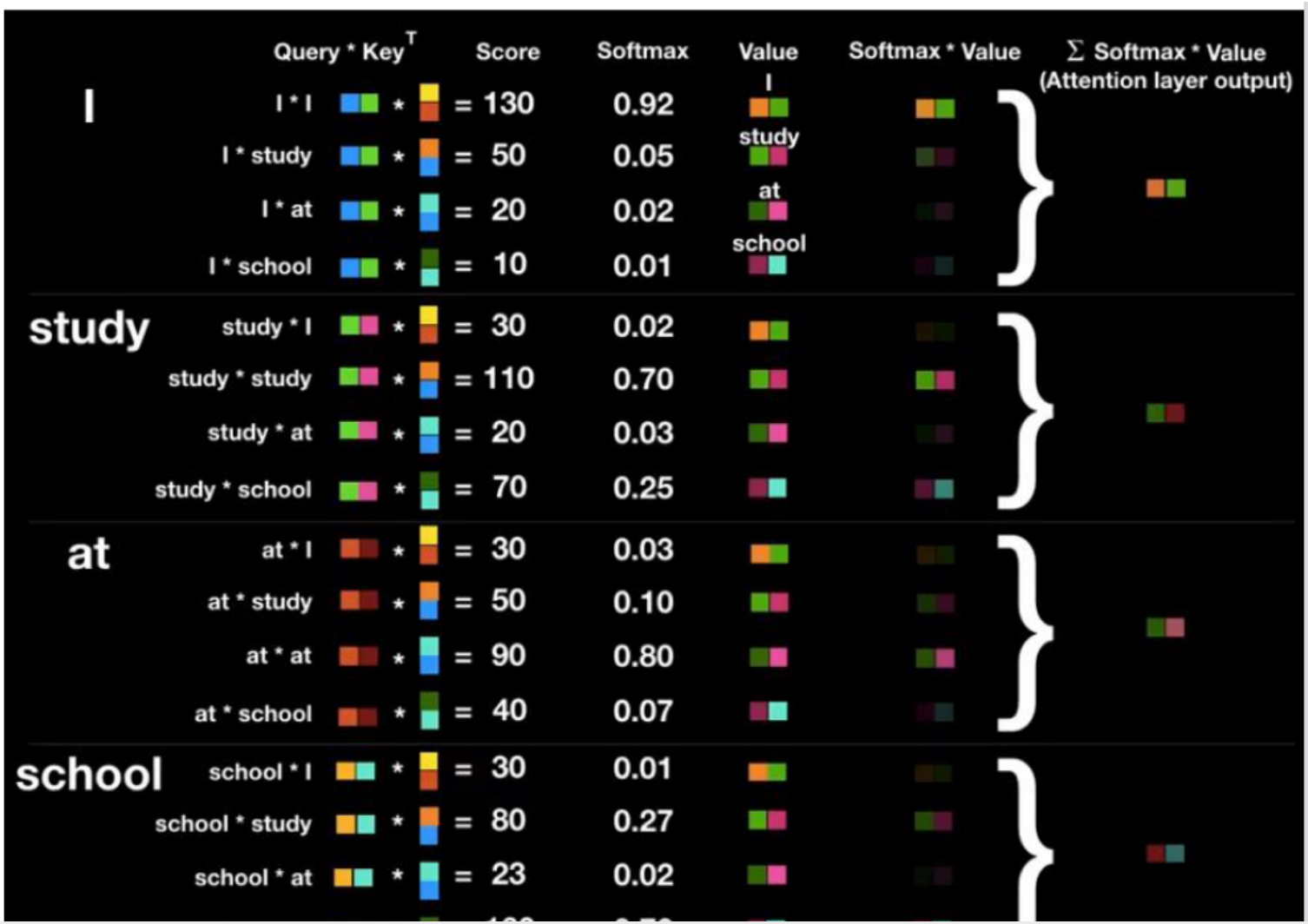
- В итоге получаем:

$$\begin{aligned} H' &= A_{L\times L} V_{L\times d}, \\ A &= \mathrm{softmax}(Q_{L\times d} K^T_{L\times d}), \\ Q &= g(H_{L\times d} W^{query}_{d\times d}), \\ V &= g(H_{L\times d} W^{value}_{d\times d}), \\ K &= g(H_{L\times d} W^{key}_{d\times d}) \end{aligned}$$

Механизм самовнимания: иллюстрация

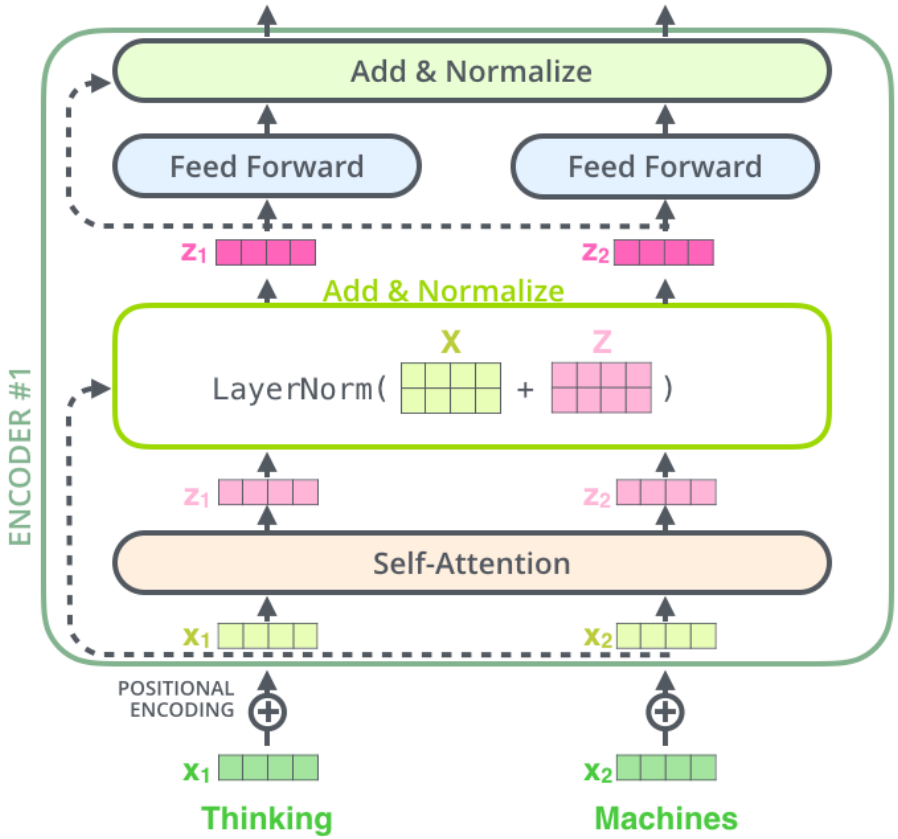


Механизм самовнимания: иллюстрация



Механизм самовнимания: трансформеры

- Механизм внимания – это один слой трансформерной архитектуры.
- Между такими слоями вставляются residual-переходыполносвязные подслои и слой-нормализация (LayerNorm). Параллельно с трансформерным блоком вставляется residual-переход.



Механизм самовнимания: множественное внимание

- Может возникнуть потребность проявлять внимание с точки зрения разных аспектов и к разным словам.

Вася съел большую банку варенья.

банку $\rightarrow$ большую морфология

банку $\rightarrow$ съел семантика

Механизм самовнимания: множественное внимание

- Может возникнуть потребность проявлять внимание с точки зрения разных аспектов и к разным словам.

Вася съел большую банку варенья.

банку → большую морфология

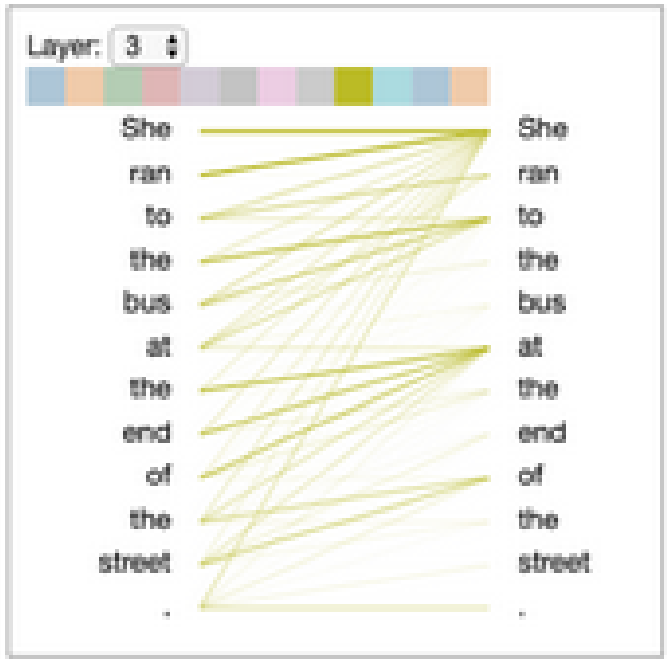
банку → съел семантика

- Это делается с помощью множественного внимания (multi-head attention):

$$\begin{aligned} H'_i &= \mathrm{Attention}(HW_{q,i}, HW_{k,i}, HW_{v,i}) \\ H' &= \mathrm{Concat}(H'_1, \ldots, H'_m) \\ W_{\star,i} &\in \mathbb{R}^{D \times \frac{D}{m}} \end{aligned}$$

- Все эти операции можно делать параллельно.

Механизм самовнимания: множественное внимание



Механизм самовнимания: энкодер-декодер

- В энкодере механизм внимания нужен, чтобы пересчитывать состояния модели, кодирующие исходное предложение.
- В декодере нужно учитывать не только исходное предложение, но и сгенерированную часть результата.

Механизм самовнимания: энкодер-декодер

- В энкодере механизм внимания нужен, чтобы пересчитывать состояния модели, кодирующие исходное предложение.
- В декодере нужно учитывать не только исходное предложение, но и сгенерированную часть результата.
- Как следствие, есть два подслоя внимания:
  - Внимание состояний энкодера к состояниям декодера α_{ij}:

$$i \quad - \quad \text{позиция в генерируемом тексте,}$$
$$j \quad - \quad \text{позиция в исходном тексте.}$$

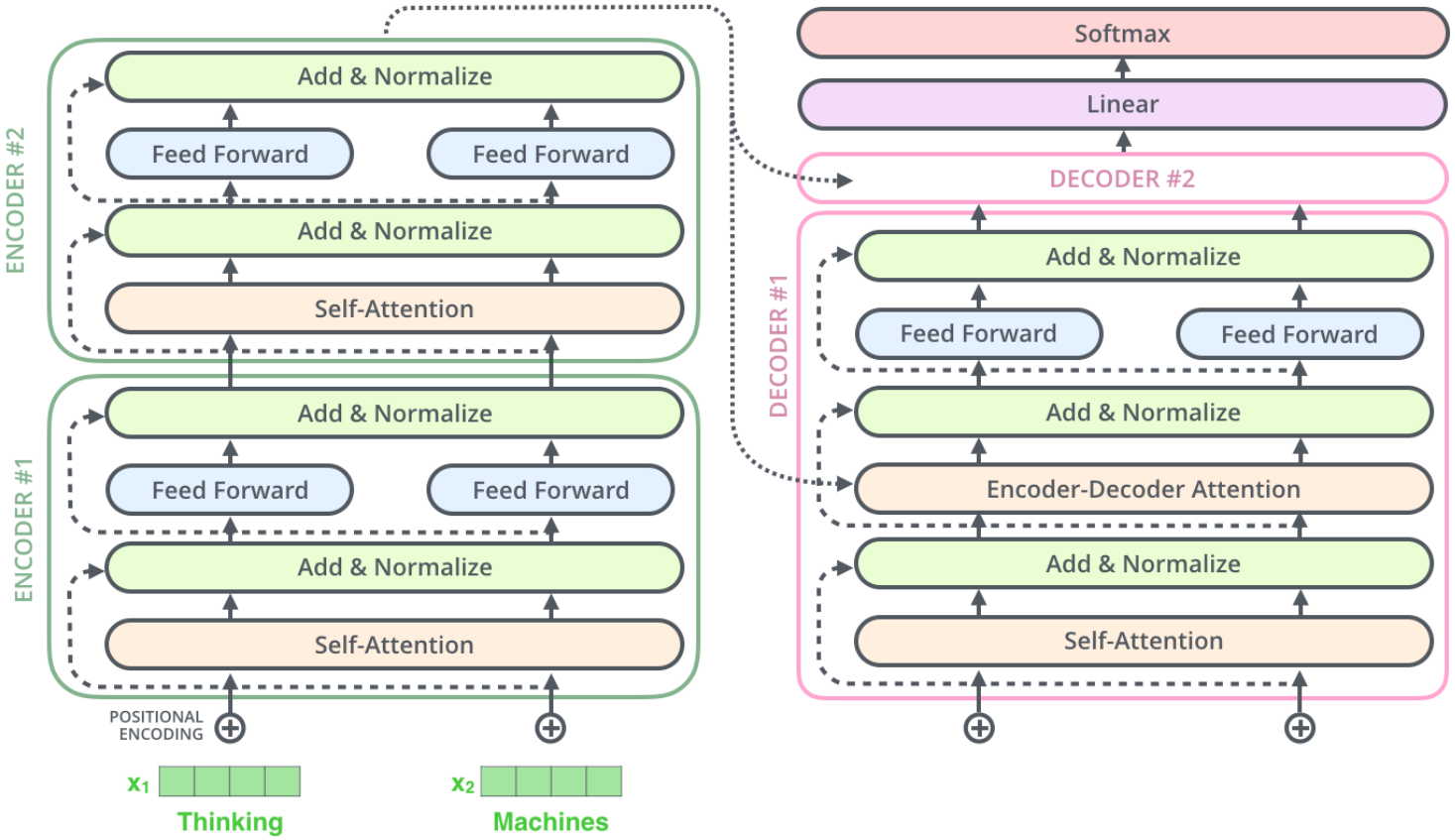
Механизм самовнимания: энкодер-декодер

- В энкодере механизм внимания нужен, чтобы пересчитывать состояния модели, кодирующие исходное предложение.
- В декодере нужно учитывать не только исходное предложение, но и сгенерированную часть результата.
- Как следствие, есть два подслоя внимания:
  - Внимание состояний энкодера к состояниям декодера α_{ij}:

    i – позиция в генерируемом тексте,
    j – позиция в исходном тексте.
  - Внимание состояний энкодера к состояниям декодера β_{ij}:

    i – позиция в генерируемом тексте,
    j – позиция в генерируемом тексте, $j < i$.
  - При обучении считается $\beta_{ij} = 0$ при $j \geqslant i$, чтобы модель не заглядывала в будущее.

Трансформеры: энкодер-декодер



Трансформеры: скорость

- Скорость передачи информации на m шагов:
  - Свёрточные сети: $O(\frac{m}{w})$.
  - Рекуррентные сети: $O(m)$.
  - Трансформеры: $O(1)$.
- Затраты памяти на последовательность длины L:
  - Свёрточные сети: $O(wd^2L)$.
  - Рекуррентные сети: $O(d^2L)$.
  - Трансформеры: $O(L^2d)$.

Трансформеры: позиционное кодирование

- Пока механизм внимания никак не различает одинаковые вектора, стоящие в разных местах.
- Чтобы это исправить, конкатенируют вектора слов с позиционными эмбеддингами:

$$x_i = [emb(word_i), x_{pos}(i)]$$

Трансформеры: позиционное кодирование

- Пока механизм внимания никак не различает одинаковые вектора, стоящие в разных местах.
- Чтобы это исправить, конкатенируют вектора слов с позиционными эмбеддингами:

$$x_i = [emb(word_i), x_{pos}(i)]$$

- x_{pos} иногда задают явно (Vaswani et al., 2017):

$$\begin{aligned}(x_{pos}(i))_{2j} &= \sin \frac{i}{10000^{j/d}}, \\ (x_{pos}(i))_{2j+1} &= \cos \frac{i}{10000^{j/d}}.\end{aligned}$$

- В более поздних подходах их сделали обучаемыми:
  - Это дополнительная матрица размера max_length $\times$ d.
  - Последовательности длиннее max_length не обрабатываются.