

Глубокое обучение и вообще

Бекезин Никита

1 декабря 2021 г.

Лекция 14: Быстрое введение в SOTA

Agenda

- seq2seq
- Attention
- ELMO
- Fasttext

задача seq2seq

спойлер

После этой лекции могут возникнуть огромное количество вопросов - но в современных архитектурах слишком много инженерных хаков, которые лучше осознавать постепенно сами.

Я буду оставлять некоторые ключевые слова того, чтобы вы могли сами залезть поглубже, если такое погружение потребуется.

Задача seq2seq - задача, когда мы хотим предсказать по одной последовательности другую Самая стандартная подобная задача - машинный перевод. Нейронные сети ворвались в эту сферу человеческого прогресса в 2014 году

Метрика

Модели в машинном переводе сравнивают по BLEU score - если в кратце, то эта метрика сравнения полученного машиной перевода и человеческого, насколько мы вообще бьемся. Из проблем данной метрики - если машина перевела правильно, но альтернативно, то BLEU будет низкий....

BLEU

$$BLEU = \text{brevity penalty} \cdot \left(\prod_{i=1}^n \text{precision}_i \right)^{1/n} \cdot 100\%$$

$$\text{brevity penalty} = \min \left(1, \frac{\text{output length}}{\text{reference length}} \right)$$

BLEU

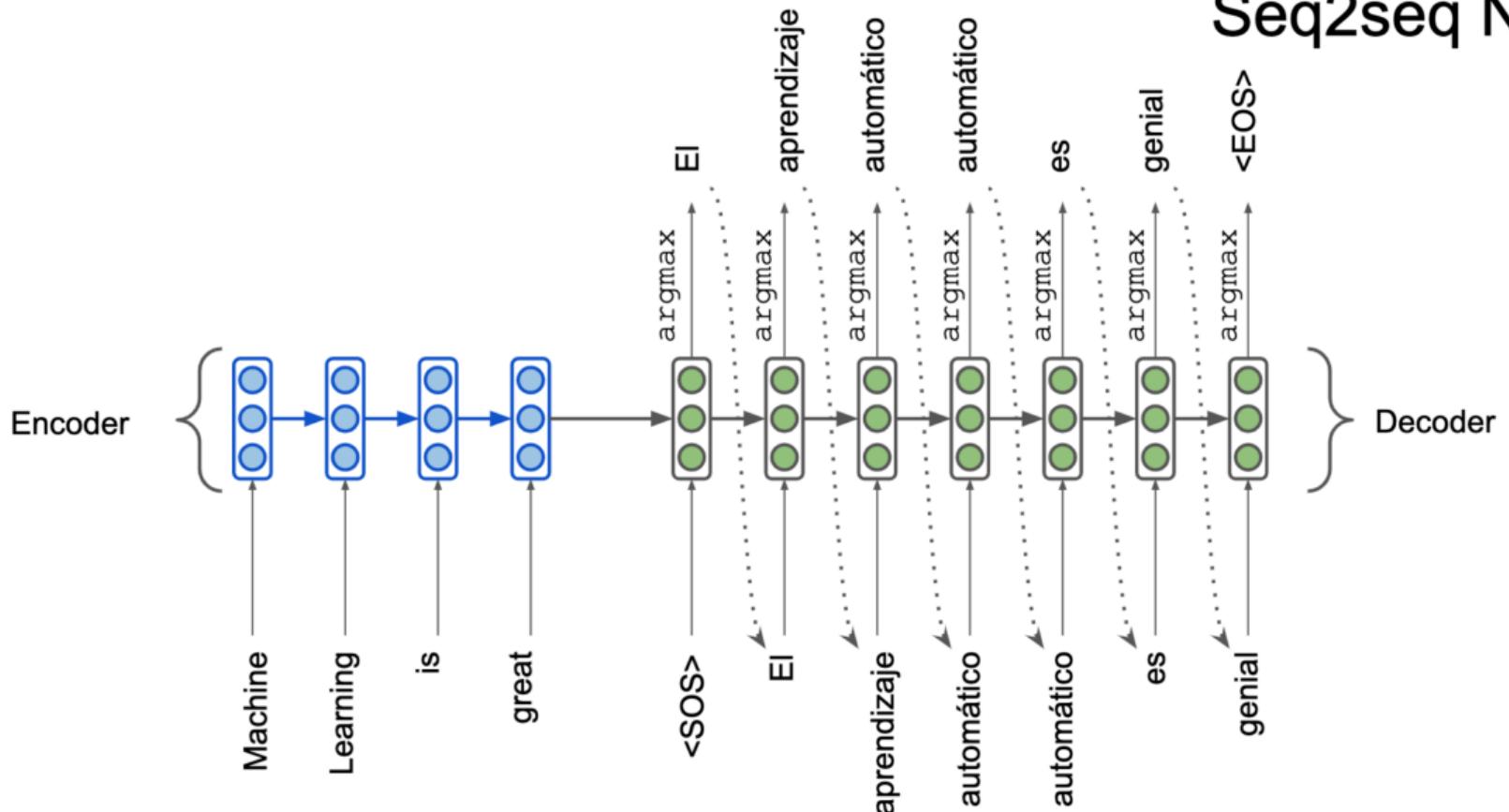
SYSTEM A: **Israeli officials** responsibility of **airport** safety
2-GRAM MATCH 1-GRAM MATCH

REFERENCE: Israeli officials are responsible for airport security

SYSTEM B: **airport security** **Israeli officials are responsible**
2-GRAM MATCH 4-GRAM MATCH

Metric	System A	System B
precision (1gram)	3/6	6/6
precision (2gram)	1/5	4/5
precision (3gram)	0/4	2/4
precision (4gram)	0/3	1/3
brevity penalty	6/7	6/7
BLEU	0%	52%

Seq2seq NMT

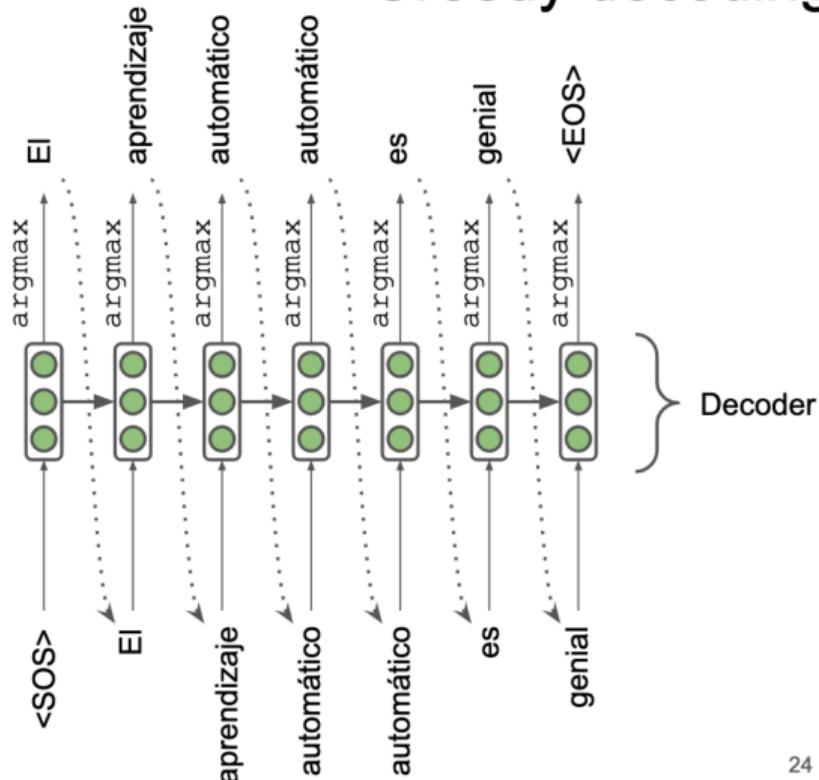


- Decoder predicts the most probable token (argmax) on each step
- The approach is **greedy**

Any problems with it?

Any mistake is treated as input on the next step!

Greedy decoding



Greedy decoding

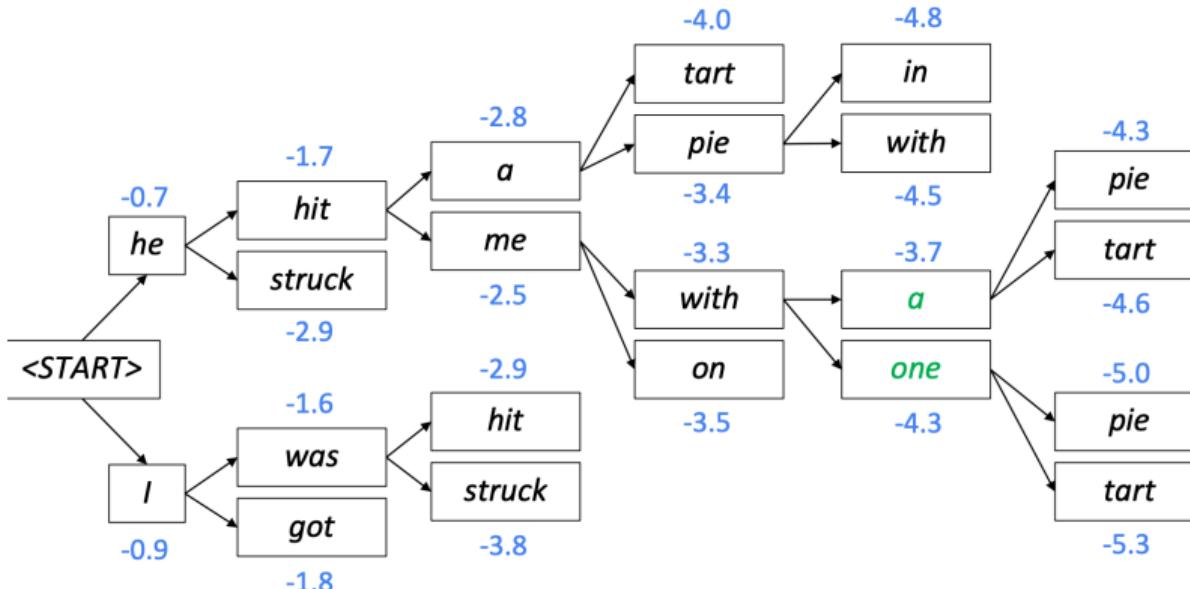
При декодировании может быть следующие проблемы - мы декодируем какое-то конкретное слово. Но что делать если это слово некоректное?

Greedy decoding/beam search

Самый простой подход - селектить несколько наиболее вероятных слов, а не одно. Мы получим множество предложений, а потом по какой-то эвристике выбирать лучшее из них. Подход хорош всем, кроме скорости.
Ему есть альтернатива - называется beam-search.

Beam search decoding: example

Beam size = $k = 2$. Blue numbers = $\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$



For each of the k hypotheses, find top k next words and calculate scores

Attention!

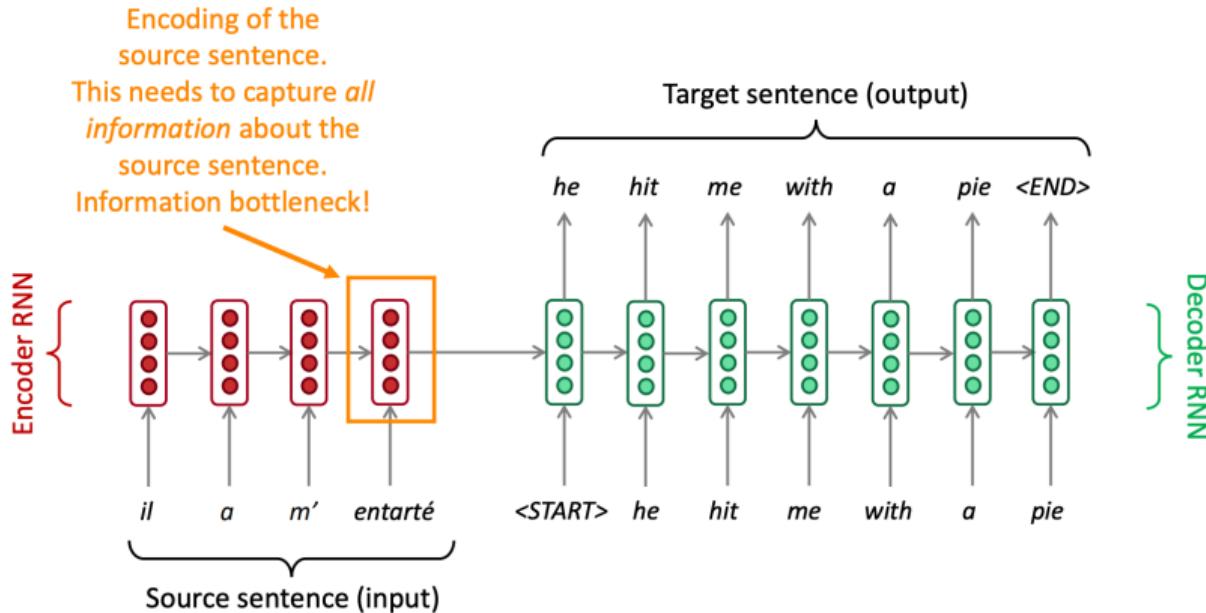
Attention

А вот бы использовать не один вектор, а все. Информация то течет и кодируется во всех векторах.....

И да - это классная и разумная идея. Нам на встречу приходит концепция внимания.

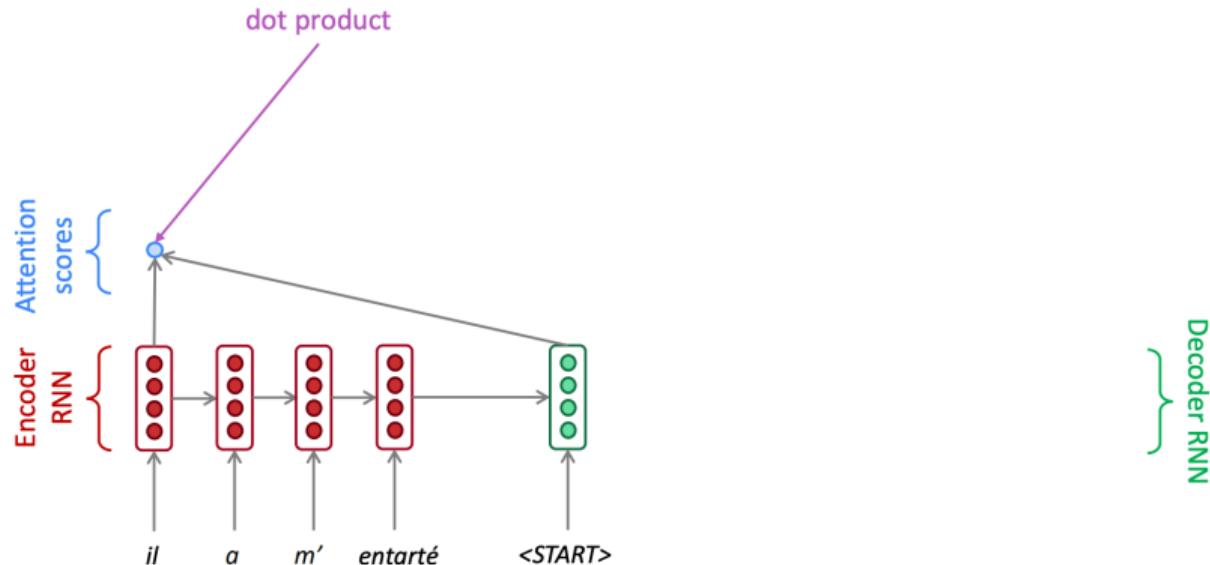
Attention

Sequence-to-sequence: the bottleneck problem



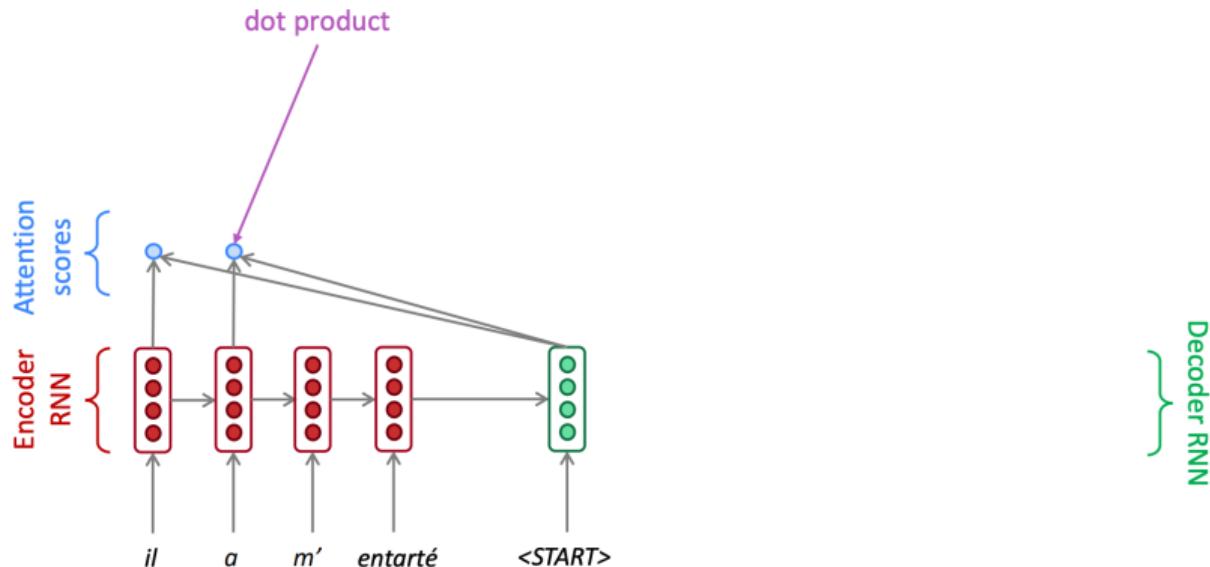
Attention

Sequence-to-sequence with attention



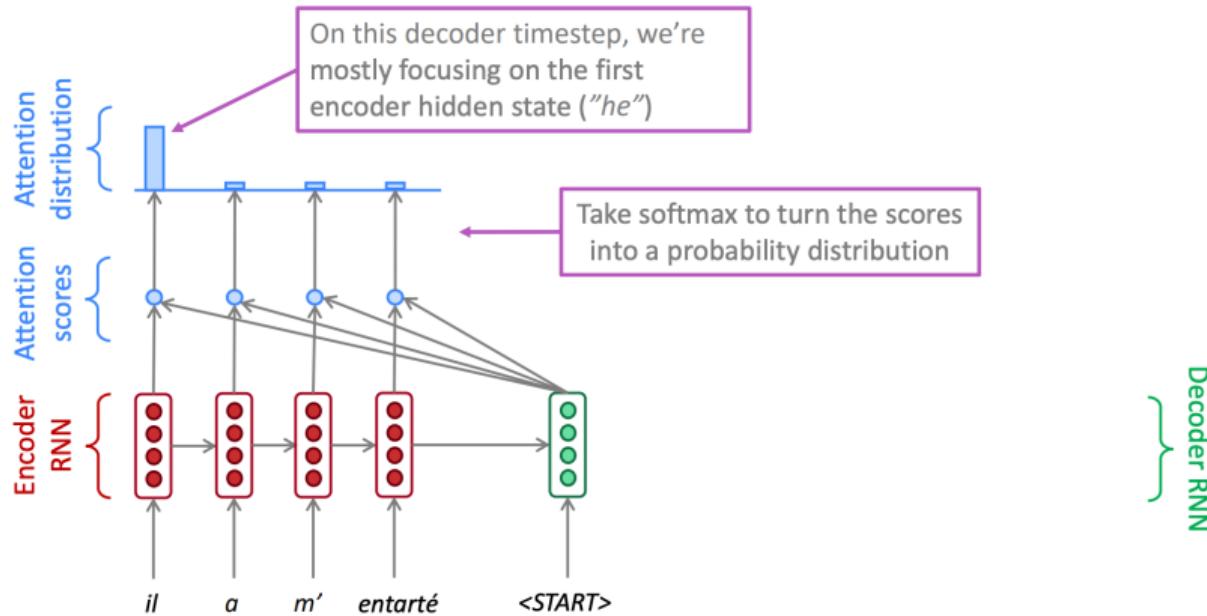
Attention

Sequence-to-sequence with attention



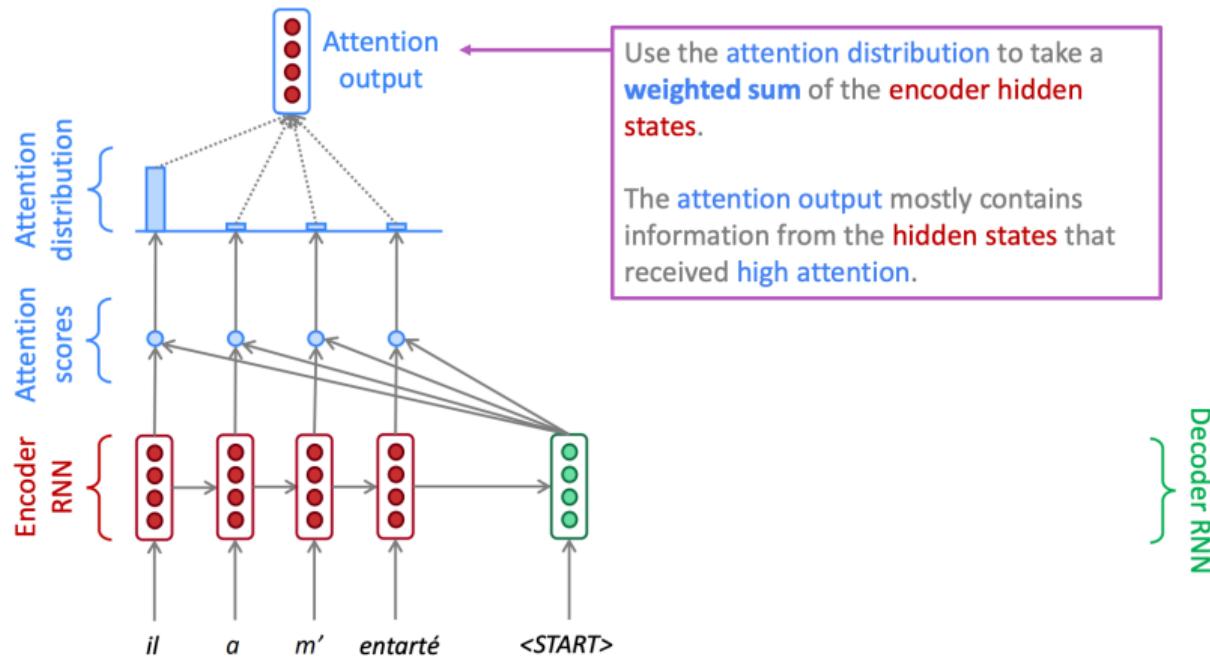
Attention

Sequence-to-sequence with attention



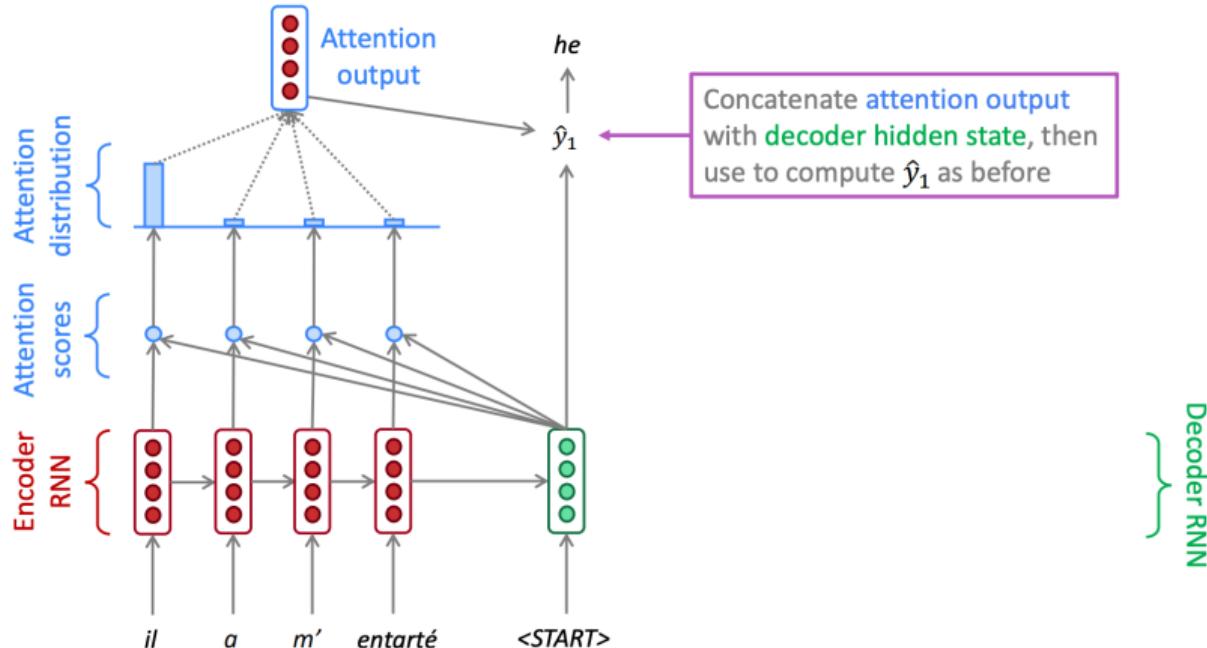
Attention

Sequence-to-sequence with attention



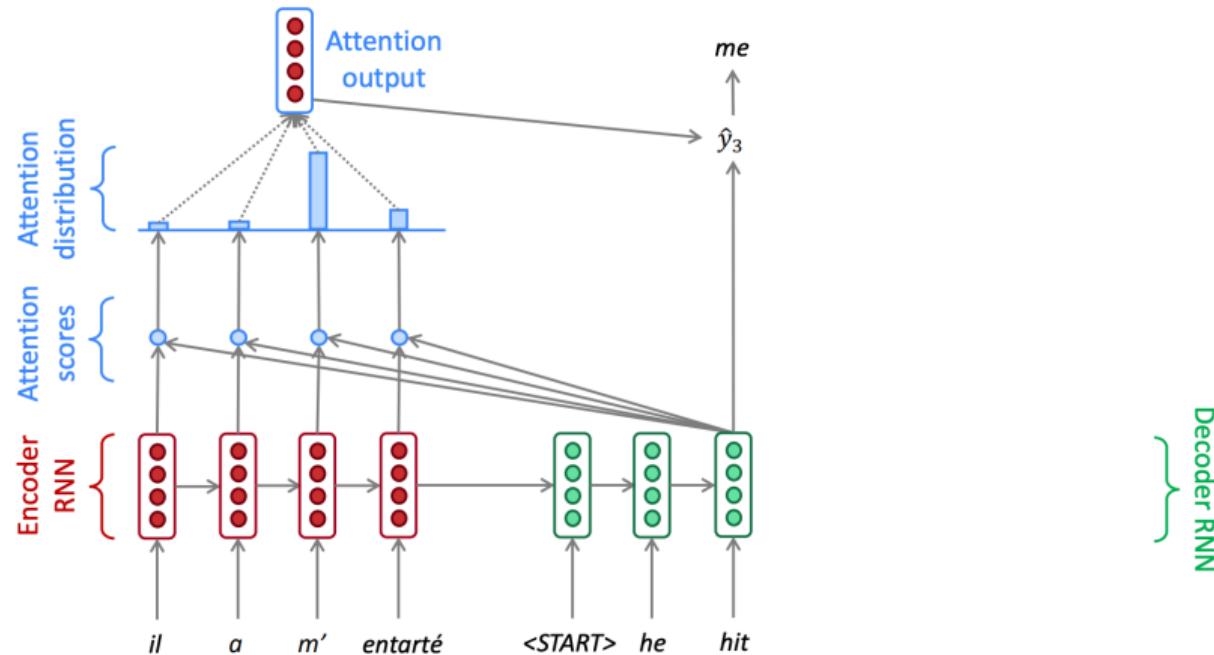
Attention

Sequence-to-sequence with attention



Attention

Sequence-to-sequence with attention

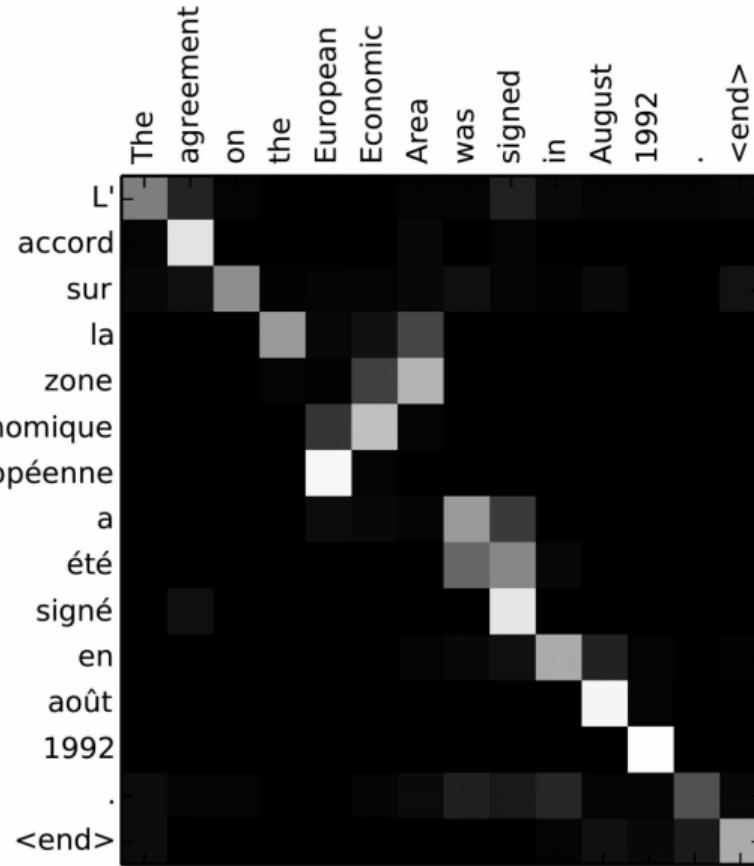


Attention

There are **several ways** you can compute $e \in \mathbb{R}^N$ from $\mathbf{h}_1, \dots, \mathbf{h}_N \in \mathbb{R}^{d_1}$ and $s \in \mathbb{R}^{d_2}$:

- Basic dot-product attention: $e_i = s^T \mathbf{h}_i \in \mathbb{R}$
 - Note: this assumes $d_1 = d_2$
 - This is the version we saw earlier
- Multiplicative attention: $e_i = s^T \mathbf{W} \mathbf{h}_i \in \mathbb{R}$
 - Where $\mathbf{W} \in \mathbb{R}^{d_2 \times d_1}$ is a weight matrix
- Additive attention: $e_i = \mathbf{v}^T \tanh(\mathbf{W}_1 \mathbf{h}_i + \mathbf{W}_2 s) \in \mathbb{R}$
 - Where $\mathbf{W}_1 \in \mathbb{R}^{d_3 \times d_1}$, $\mathbf{W}_2 \in \mathbb{R}^{d_3 \times d_2}$ are weight matrices and $\mathbf{v} \in \mathbb{R}^{d_3}$ is a weight vector.
 - d_3 (the attention dimensionality) is a hyperparameter

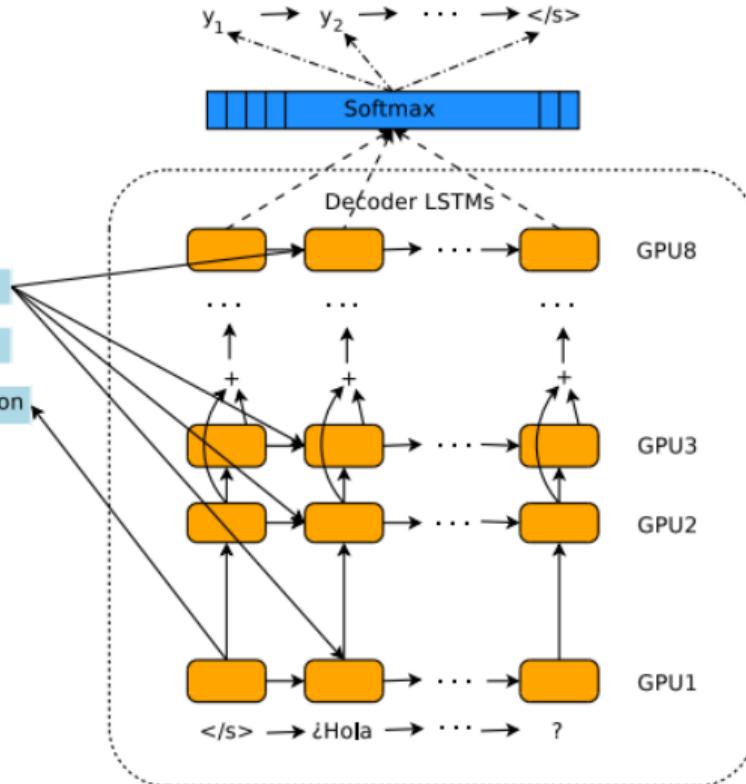
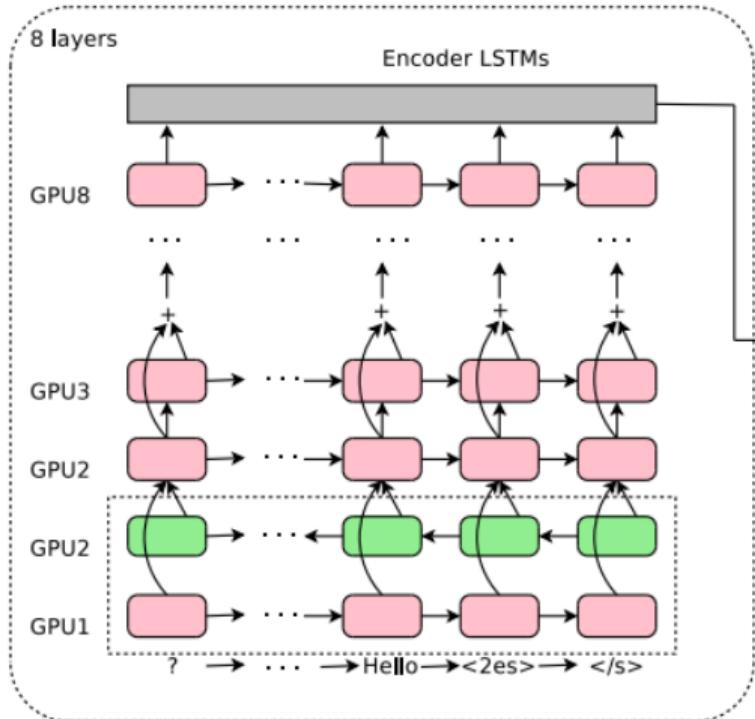
Attention



Attention

Идейно - внимание просто выбирает то из эмбедингов, которое действительно нужно для декодирования. Это просто матричное произведение(а можно взвешивать и без весов) и softmax. У нас все остается дифференцируемым - берем градиентны, накапливаем инфу в весах сетки.

google



В целом глобальное решение было найдено, осталось закидать проблему железом.

Выводы:

1. 8 слоев LSTM (8 Карл!)
2. в attention 2 слоя dense.
3. Собираем слова из морфем - пытаемся победить out-of-vocabulary.
4. Модель стала порой нетактичной - требуются слишком большие дата сеты, чтобы учить эту большую прелесть.

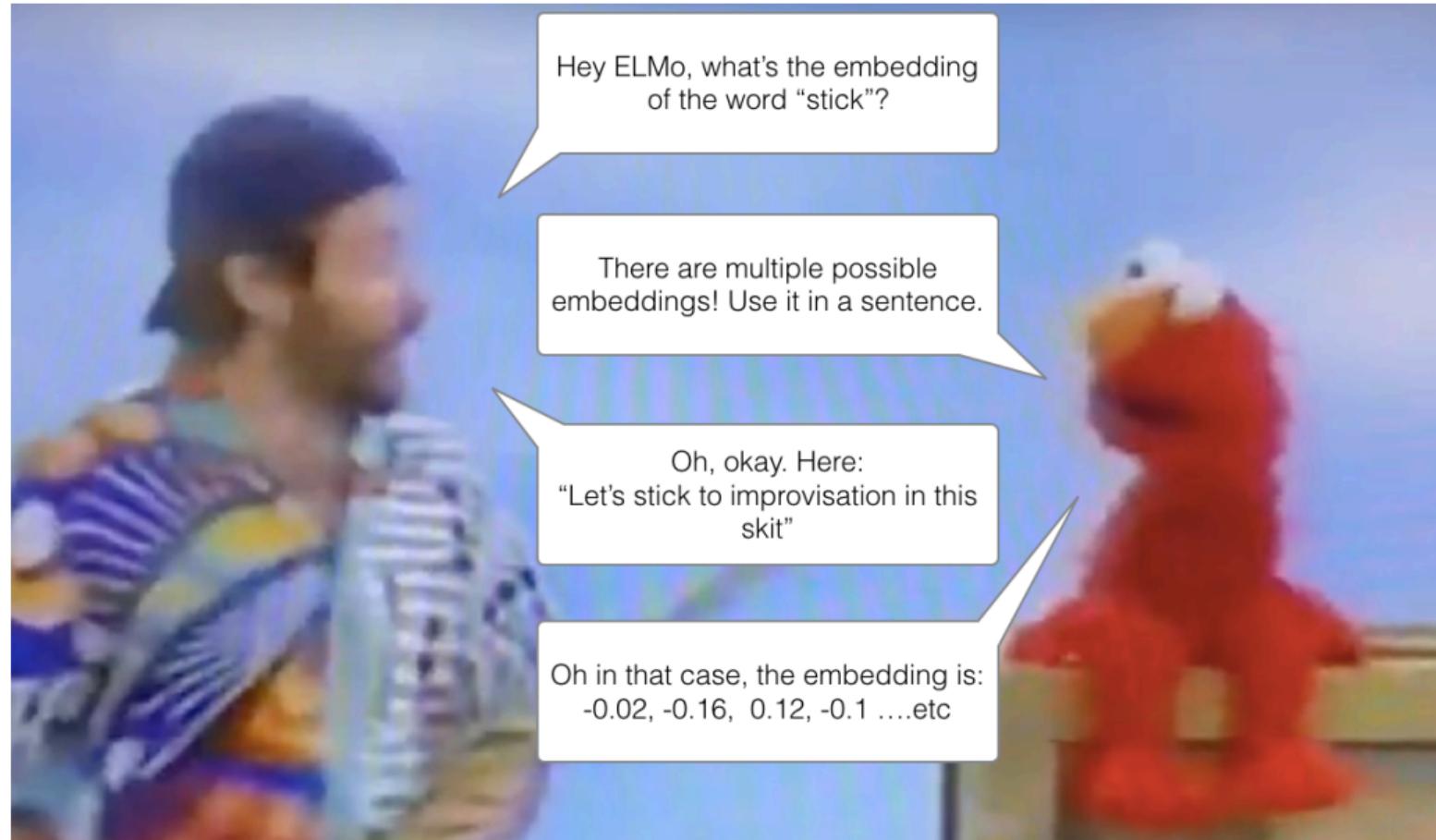
ELMO

Серия вопросов в зал

Как работают разные эмбединги?

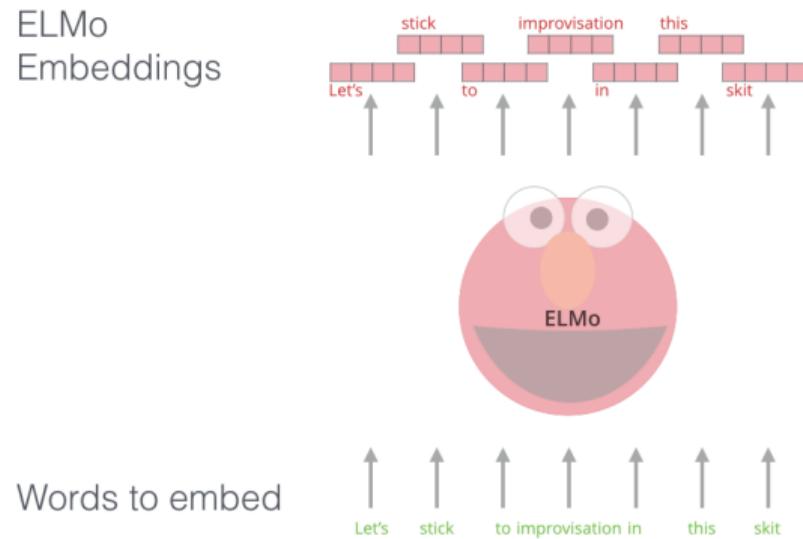
В чем, по вашему мнению, их главная проблема?

Картиночка про решение проблемы



ELMO

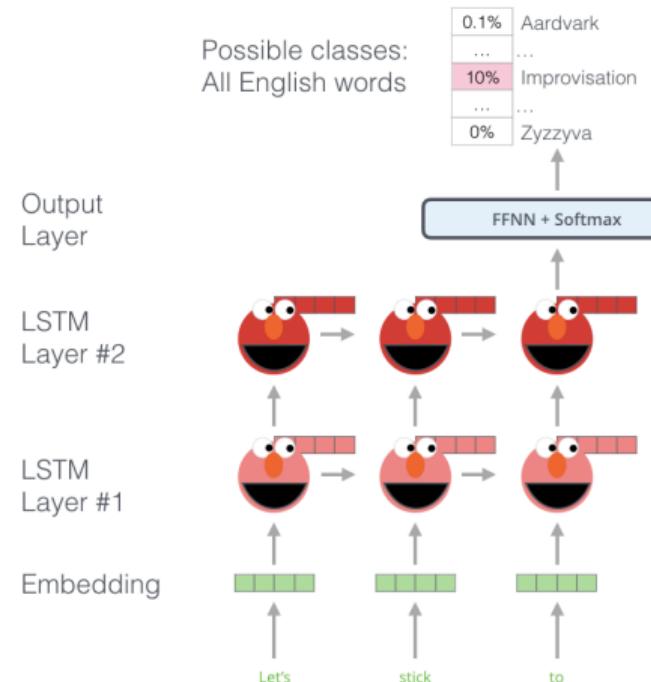
Захватываем контекст предложения через biderictional LSTM. Таким образом мы захватываем и контекст предложения (да, надо очень очень много данных, не обучайте это дома)



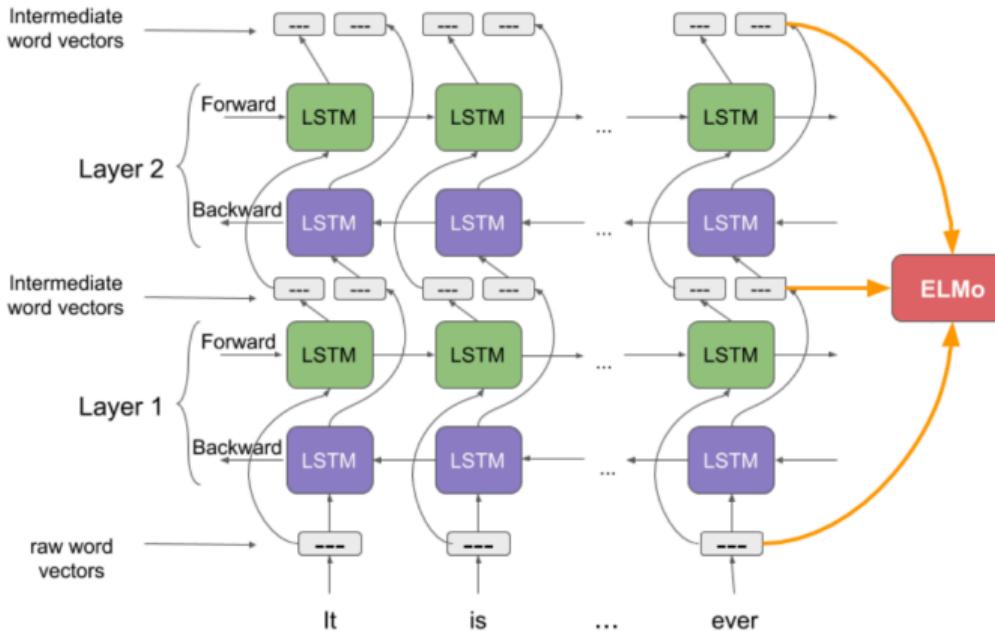
Рисунки на слайдах заимствованы [отсюда](#)

ELMO

Учится понимать язык ELMO следующим образом - оно берет большой дата сет и пытается предсказать следующее слово в предложении (Language Modeling task).



Архитектура ELMO



Source

ELMO

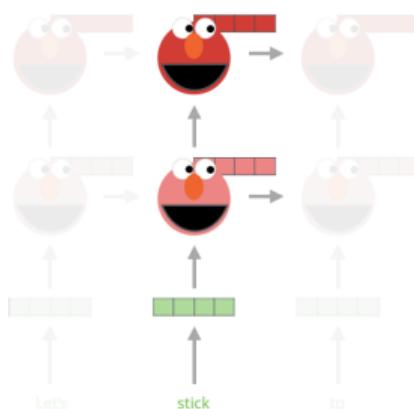
Применяем

Embedding of “stick” in “Let’s stick to” - Step #2

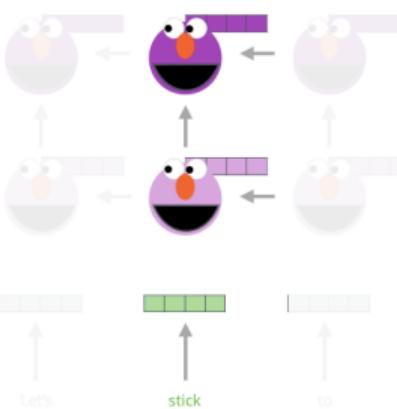
1- Concatenate hidden layers



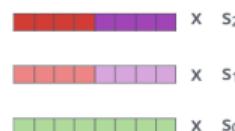
Forward Language Model



Backward Language Model



2- Multiply each vector by a weight based on the task



3- Sum the (now weighted) vectors



ELMo embedding of “stick” for this task in this context

Fasttext

Fasttext

- Усовершенствование word2vec дополнительными токенами, отвечающими за буквенные n-граммы, это решает проблему OOV (out of the vocabulary)
- Например, триграммы для where $\Rightarrow <\text{wh}, \text{whe}, \text{her}, \text{ere}, \text{re}>$
- Символы < и > это спец-символы, говорящие о том, что слово кончилось или началось
- При расчёте для слова итогового вектора будем усреднять эмбединг для слова и эмбединги для n-грамм

Важный момент: последовательность $<\text{her}>$, соответствующая слову her, отличается от 3-граммы her, полученной из слова where.

Итого

Почему это все стало круто и популярно?

1. Идея с вниманием стала ключевой - таким образом мы можем тянуть информацию через всю последовательность
2. Внимание можно параллелизовать, LSTM намного сложнее
3. Придумали как сделать так, чтобы не размечать данные
4. Купили много GPU
5. Посадили кучу инженеров, которые заставили это все учиться!

Ну, и маленькая мысль в конце - мы разобрали кубики нейронных сетей и посмотрели какие запутанные архитектуры можно из этих кубиков делать. Современный моделист в нейронных сетях - скорее инженер, нежели аналитик