

Глубокое обучение и вообще

Кузнецов Максим, Шигапова Фирюза и Соловей Влад

31 марта 2021 г.

Посиделка N: Attention и Self-attention

Agenda

- seq2seq
- Attention
- Self-attention
- BERT
- ELMO
- Сломанный мозг.....

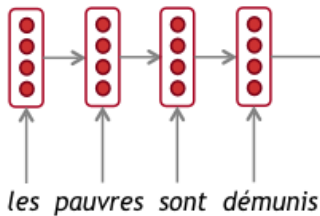
задача seq2seq

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

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

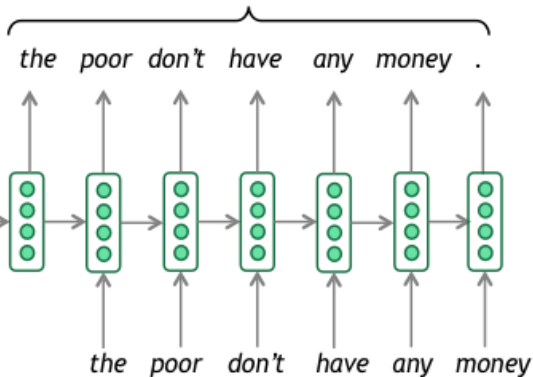
Задача seq2seq - задача, когда мы хотим предсказать по одной последовательности другую Самая стандартная подобная задача - машинный перевод. Нейронные сети ворвались в эту сферу человеческого прогресса в 2014 году

We feed in each word from left to right, one at a time. By the end, the NMT system has encoded information about the whole sentence in a numerical format.



French sentence (input)

English translation (output)



The previous outputted word gets added as part of the input into the network next, giving the network some view of the sentence already produced and some context of the words preceding it.

Как мы можем учитывать контекст?

Рекуррентные нейросети:

1. Последовательное вычисление
2. Сложность учёта зависимости длины $n - O(n)$

Сверточные нейросети:

1. Параллельное вычисление
2. Длина учитываемой последовательности зависит от глубины сети (k)
3. Непрерывные свертки дают сложность $O(n/k)$
4. Если свертки с прореживанием (dilated), то сложность $O(\log_k n)$

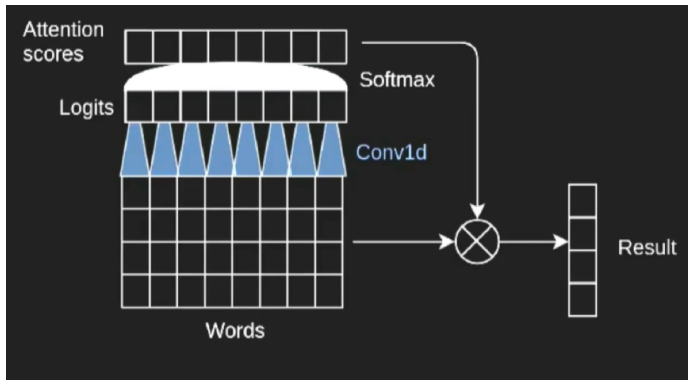
Чем плохо?

1. Медленно учится
2. Теряется информация о далеких зависимостях
3. Нужен другой вариант!

Attention!

Самый простой вариант Attention

Слов много и каждое слово хочет привлечь к себе «внимание». В классических вариантах мы одновременно выделяем важные фичи и оцениваем информацию по ним. Хочется разделить эти две части. Матрица внизу: представления слов, строка – слово. Вместо Conv1d можно использовать Linear с одним выходом для каждой размерности эмбединга.

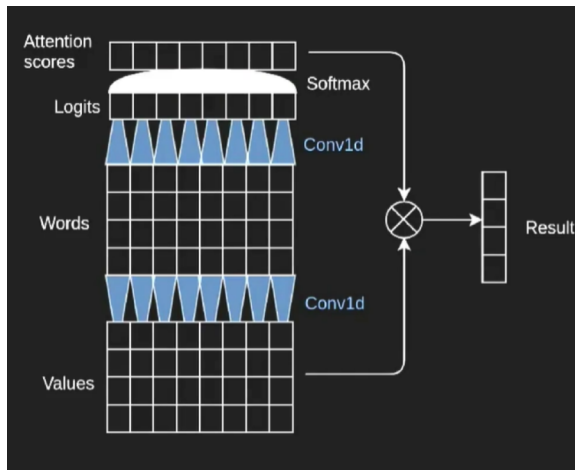


Что дает нам Attention?

1. С ростом последовательности мы ничего не теряем и не забываем.
2. Большая гибкость по сравнению с агрегациями (свертка, пулинг).
3. Значения результирующего вектора вычисляются с учетом друг друга.
4. Универсальная операция (может заменить и свертки и рекуррентные слои).
5. Возможность интерпретации.

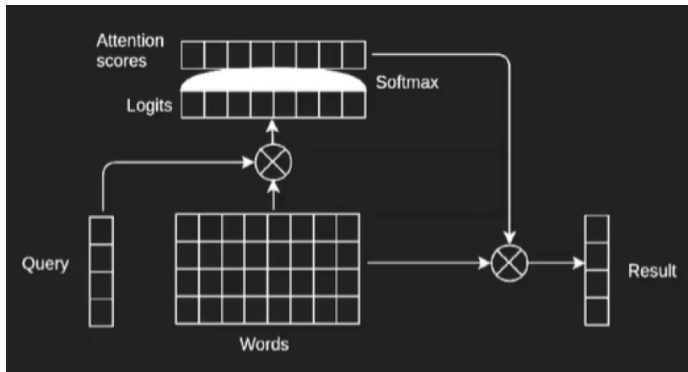
Немного усложнений

Для получения результирующего вектора можно использовать не исходные вектора, а сначала мы их немного преобразуем. Веса для получения значений (Values) и оценки важности отличаются – гибкость.



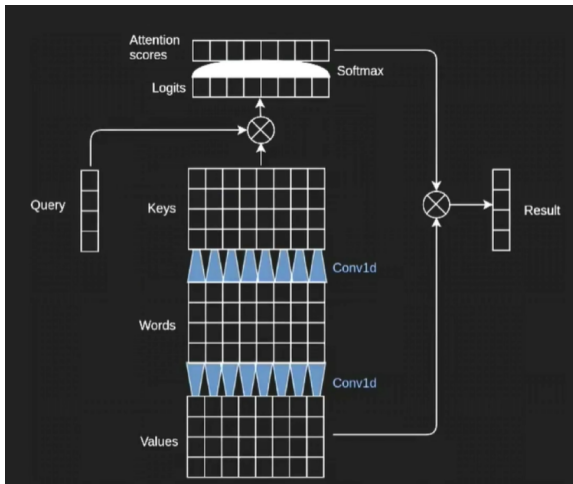
Немного усложнений

А что если нам нужно оценить релевантность относительно какого-то запроса (Query)? Например, оценка релевантности ответа на запрос пользователя, т.е. релевантность текстов.



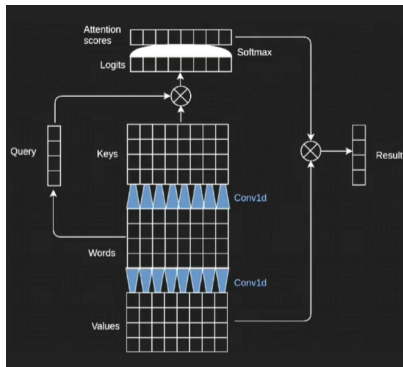
И еще немного усложнений

Еще усложним, скомбинируем оба варианта. Зачем? Еще больше гибкости и больше возможностей. Мы получаем нелинейность в определении релевантности.



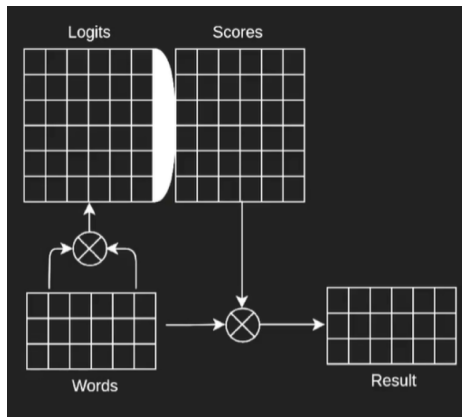
Self attention

А если нет у нас внешнего запроса? Self-Attention! Делаем Global MaxPooling для вычисления Query! Тогда оценка важности каждого слова будет обобщена на весь текст, мы учитываем больше! Т.е. мы используем глобальный контекст. Такая конструкция уже вполне может соревноваться с рекуррентками.



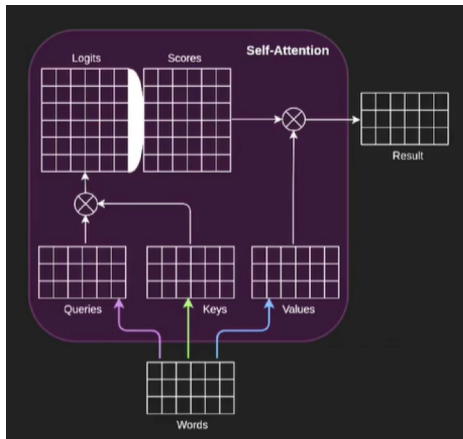
Self attention

В прошлых вариантах мы получали 1 вектор, что является сжатием. Как быть, если хочется оценивать важность каждого слова в контексте всех остальных?



Self attention прокаченный

Мы преобразовываем исходные данные при помощи линейных операций. Токен будет иметь разный вес в зависимости от того, выступает он в качестве запроса или в качестве ключа.

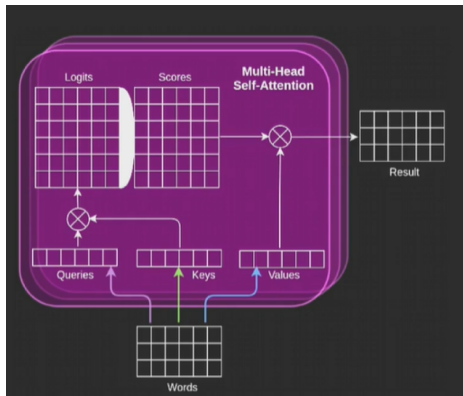


Недостатки

1. В конце операция хоть и взвешенного усреднения, но это все равно потеря информации
2. Если какой-то токен получил большой вес, то все остальные меньше => мы оценили только один аспект, хотя слова могут быть похожи разными способами (лексически, грамматически).

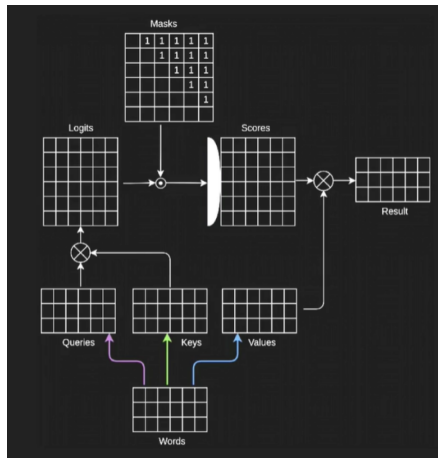
Multi-Head Self-Attention

Авторы трансформера заметили проблемы Self-Attention и предложили ввести несколько голов. В каждой голове размер признаков меньше, чем размер всей матрицы признаков. Они конкатенируются. Спойлер: это относится к Encoder в трансформере.



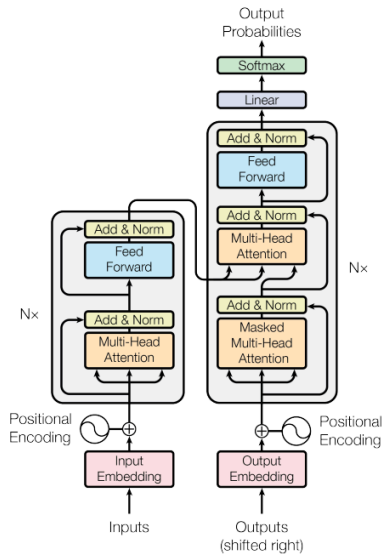
Masked Self-Attention

зрешаем нейросети видеть только те слова, что были слева от текущего.
Спойлер: это относится к Decoder в трансформере.

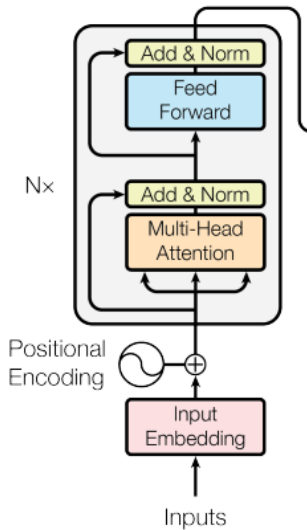


Откуда что взялось

attention is all you need

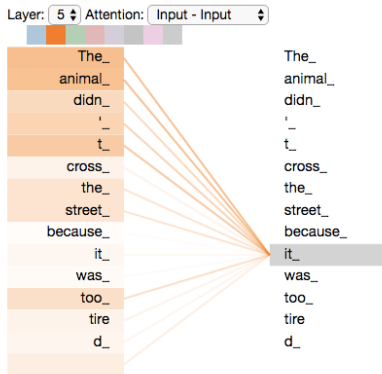


Encoder



Что мы хотим?

Есть предложение: "The animal didn't cross the street because it was too tired"



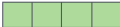
Абстракции!

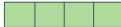
Input

Thinking


Machines


Embedding

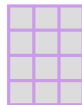
x_1 

x_2 

Queries


q_1 


q_2 



W^Q

Keys

k_1 

k_2 



W^K

Values

v_1 

v_2 



W^V

А теперь тоже самое, но словами:

1. Query, key - ищем связи между словами. Ходим по всем со всеми смотрим насколько они связаны. Query - мое текущее слово, key - мое слово с которым я сравниваю себя.
2. Value - то, что мы знаем об этом слове

Mar 2

Input

Embedding

Queries

Keys

Values

Score

Thinking

x_1 

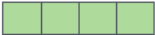
q_1 

k_1 

v_1 

$$q_1 \cdot k_1 = 112$$

Machines

x_2 

q_2 

k_2 

v_2 

$$q_1 \cdot k_2 = 96$$

Mar 3

Input

Embedding

Queries

Keys

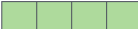
Values

Score

Divide by 8 ($\sqrt{d_k}$)

Softmax

Thinking

x_1 

q_1 

k_1 

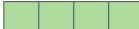
v_1 

$q_1 \cdot k_1 = 112$

14

0.88

Machines

x_2 

q_2 

k_2 

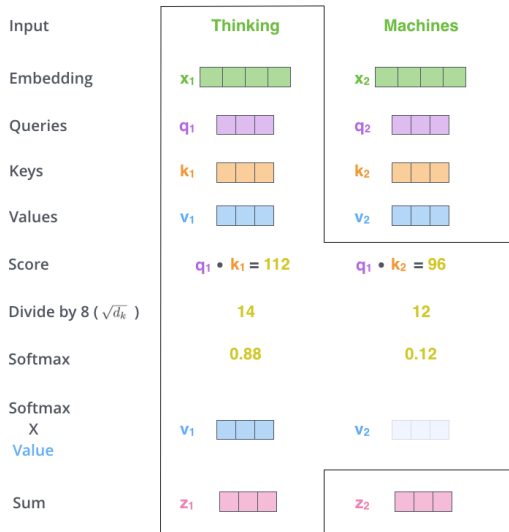
v_2 

$q_2 \cdot k_2 = 96$

12

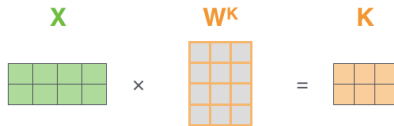
0.12

Mar 4



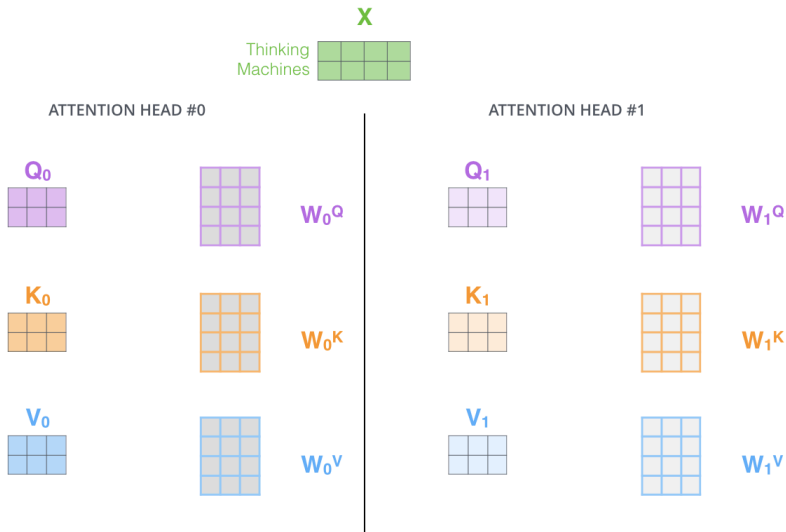
Шаг 5

$$\mathbf{X} \times \mathbf{W}^Q = \mathbf{Q}$$


$$\mathbf{X} \times \mathbf{W}^K = \mathbf{K}$$


$$\mathbf{X} \times \mathbf{W}^V = \mathbf{V}$$


multi head attention



Соединяем!

1) Concatenate all the attention heads

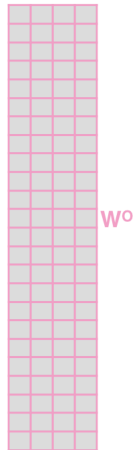


3) The result would be the Z matrix that captures information from all the attention heads. We can send this forward to the FFNN



2) Multiply with a weight matrix W^O that was trained jointly with the model

\times



Итого

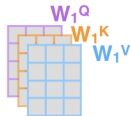
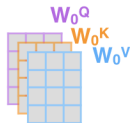
1) This is our input sentence*

Thinking
Machines

2) We embed each word*



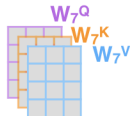
3) Split into 8 heads.
We multiply X or R with weight matrices



...

...

...



W^O

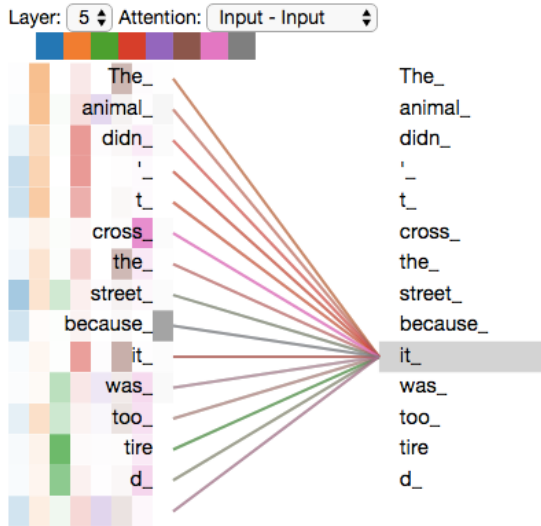


Z



* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one





Итого

Выходом всего этого дела будут вектора key и value, которые позволят декодеру смотреть на нужные нам кусочки.

Объяснение взято отсюда **английский оригинал** и отсюда **лекции мфти**

Итого

1. У нас нет никаких слоев, кроме dense
2. Учится очень классно, находит множество взаимосвязей
3. позицион энкодинг позволяет учитывать позицию в тексте



И понеслась!!! (развитие дальше - инженерные хаки и закидывание железом)

SOTA (ну или история соты)

Крутой обзорчик с техническими деталями - живут в тех же лекция физтеха. При желании можно вкурить. И да, в целом курс достаточно крутой - и крут он тем, что считается, что слушатель не лаптем щи хлебает, а считает градиент на лету, но пока не придумал зачем.

BERT

Шел 2018 год и гугл сказал - наши компьютеры самые мощные, а данные самые большие!

BERT - Bidirectional Encoder Representations from Transformers

Почему круто - придумали как предобучать без учителя (да, вот оно, вот он наш космос). И потом переиспользовать веса!

BERT

1 - **Semi-supervised** training on large amounts of text (books, wikipedia..etc).

The model is trained on a certain task that enables it to grasp patterns in language. By the end of the training process, BERT has language-processing abilities capable of empowering many models we later need to build and train in a supervised way.

Semi-supervised Learning Step

Model:



Dataset:



Objective:

Predict the masked word
(language modeling)

2 - **Supervised** training on a specific task with a labeled dataset.

Supervised Learning Step

Classifier

75% Spam
25% Not Spam

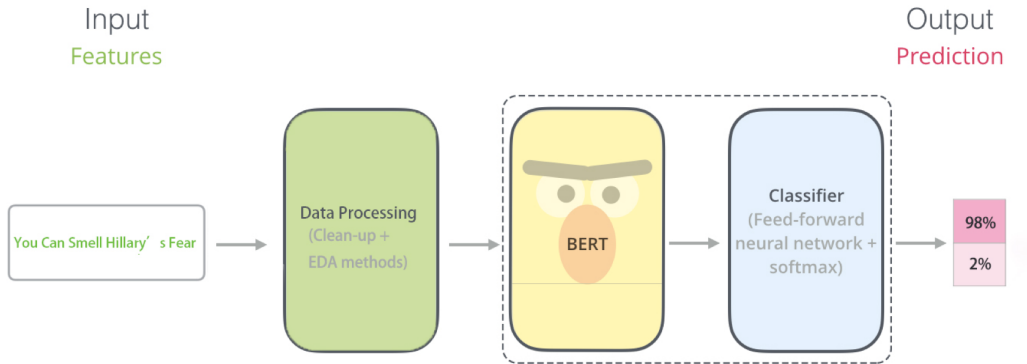
Model:
(pre-trained
in step #1)



Dataset:

Email message	Class
Buy these pills	Spam
Win cash prizes	Spam
Dear Mr. Atreides, please find attached...	Not Spam

BERT



Лежащие внутри идеи и почему он популярный

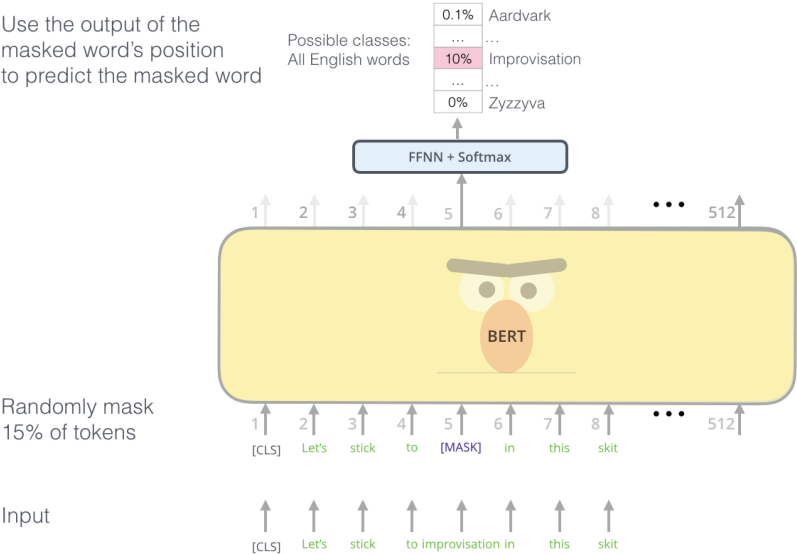
1. Предобучаем по двум задачам - берем корпус текстов и маскируем часть слов, заставляем учить и предсказывать маску.
2. И вторая идея - предсказываем следующее слово в предложении
3. Он из коробки знает язык, ему 1-2 эпохи надо подсказать, что с этим знанием делать
4. В готовых либах лежат много готовых под задачи бертов - классификация, вопросно - ответные системы и тому подобное.
5. Опять же - подаем слова кусочками, чтобы как-то решать проблему оов.

Все мы всех победили - нам не нужно размечать данные для обучения, мы счастливы!

Обзорчик на забугорном

предобучение

Use the output of the masked word's position to predict the masked word



Почему заработало?

Model Type	Vocab Size	Hidden Dim	# Params	Model Size (MB)	FLOPS ratio
BERT _{DISTILLED}	4928	48	1,775,910	6.8	1.3%
		96	5,665,926	22	1.32%
		192	19,169,094	73	4.49%
BERT _{BASE}	30522	768	110,106,428	420	100%

ELMO

Серия вопросов в зал

Как работают разные эмбединги?

В чем, по вашему мнению, их главная проблема?

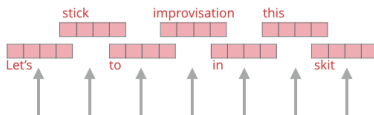
Картиночка про решение проблемы



ELMo

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

ELMo
Embeddings

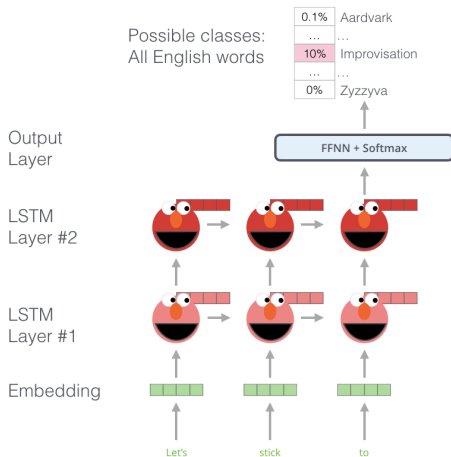


Words to embed



ELMO

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



Применяем

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

1- Concatenate hidden layers



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

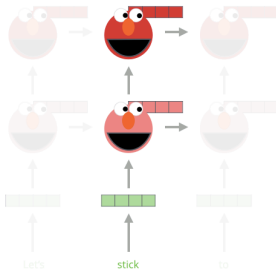


3- Sum the (now weighted) vectors

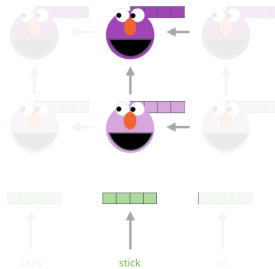


ELMo embedding of “stick” for this task in this context

Forward Language Model



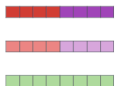
Backward Language Model



Применяем

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

1- Concatenate hidden layers



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

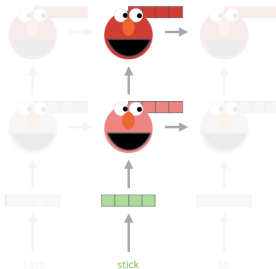
 $\times S_2$
 $\times S_1$
 $\times S_0$

3- Sum the (now weighted) vectors

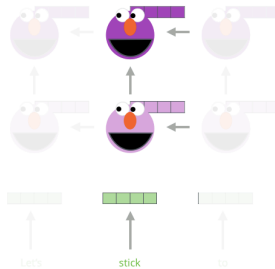


ELMo embedding of “stick” for this task in this context

Forward Language Model



Backward Language Model



Итого

Почему это все стало круто и популярно?

1. Идея с вниманием стала ключевой - таким образом мы можем тянуть информацию через всю последовательность
2. Вниманию 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 - за курс мы разобрали кубики нейронных сетей и посмотрели какой лютейший треш можно из этих кубиков делать. Современный моделист в нейронных сетях - скорее инженер, нежели аналитик