Вопросно-ответные системы

Цель занятия

После освоения темы:

- вы научитесь строить модели, которые смогут дать ответы на поставленные вопросы;
- рассмотрите, какие существуют наборы данных для решения задачи построения вопросно-ответных систем;
- узнаете, какие существуют подходы к решению задачи построения вопросно-ответных систем;
- изучите задачу построения вопросно-ответных систем в открытом домене;
- разберетесь в GPT, GPT-2, GPT-3.

План занятия

1. Задача построения вопросно-ответных систем
2. SQuAD и SberQuAD
3. Подходы к построению задачи вопросно-ответных систем
4. Вопросно-ответная система для открытого контекста
5. GPT-2 и GPT-3
6. Подробнее о GPT

Конспект занятия

1. Задача построения вопросно-ответных систем

Историческая справка

Еще в 70-е годы XX века, когда проводились эксперименты с машинным переводом, Сименс и его команда предложили неплохой обзор решений к построению вопросно-ответных систем. Их решение опиралось на поиск соответствий в некотором наборе документов. По сути, искалась некоторая мера близости между вопросом и соответствующим ему ответом.

Это решение задачи построения вопросно-ответных систем — что-то похожее на поиск. Те самые поисковики, которыми мы пользуемся сейчас, решают похожую задачу. У нас есть огромное множество документов (веб-страниц) и есть запрос (вопрос поисковику). Поисковик сортирует документы по релевантности с помощью хитрых алгоритмов и в первую очередь показывает те, что соответствуют нашему запросу (или те, что заплатили за рекламу и за поднятие в строке выдачи).

В 90-х и в нулевых люди стали подходить к задаче построения вопросно-ответных систем более серьезно. Появились и вычислительные мощности, и различные датасеты в цифровом виде. Использовалась энциклопедия, которая находилась в онлайн.

В 1999 году появился еще один подход к этой задаче — NIST TREC QA.

В 2011 году IBM представил свою систему DeepQA.

С 2014 года появляются более продвинутые системы, которые во многом обязаны не только развитию глубокого обучения, но и появлению наборов данных, которые позволяют обучаться этим системам. Например, DrQA, представленная в 2016 году, опиралась на датасет SQuAD (Stanford Question Answering Dataset) и на различные наработки.

Задача построения вопросно-ответных систем

Задача построения вопросно-ответных систем достаточно широкая. Раньше мы делали поиск ответа из набора бета-классов (как в задаче классификации), предсказания числа (как в задаче регрессии) и генерации текста (как в задаче языкового моделирования). Теперь мы должны не просто сгенерировать перевод

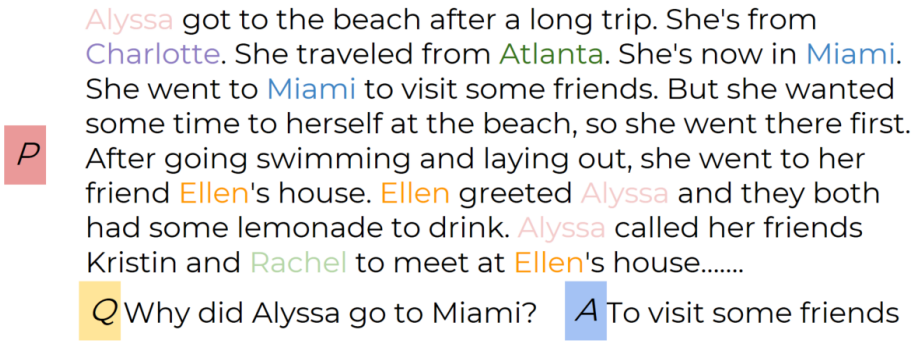
нужной фразы или предсказать метку класса для данного объекта, а найти или сгенерировать ответ для заданного вопроса. Для этого нужно как понять вопрос, так и знать, откуда взять ответ, то есть обладать достаточными знаниями. Именно поэтому разделяют Question Answering и Open-Domain Question Answering. На закрытом контексте: Context-Based и Open-Domain Question Answering.

Может быть две ситуации:

- Вам предоставляют вопрос и базу знаний. В простейшем случае — текст, набор каких-то предложений, в которых нужно найти ответ.
- Вам дают вопрос и говорят: «Ну, вперед, отвечайте на вопрос». Все, как на экзамене: смотреть никуда нельзя, книжки закрыты.

Для начала разберемся с Context-Based Question Answering. У нас есть некоторый контекст, на который мы опираемся в поиске ответа.

Пример.



Вопросно-ответная система, если она завязана на контексте, может сгенерировать фразу to visit some friends (ответ на вопрос) или выделить кусок текста и сказать: «Здесь ответ». Понятно, что выделить кусок текста несколько проще, чем сгенерировать ответ с ходу.

Наша задача — сгенерировать ответ, то есть сгенерировать некоторую строку, которая будет содержать ответ на поставленный вопрос.

Чтобы учить такие модели и решать такие задачи, нужно иметь богатый набор данных, на которых эти модели будут обучаться. Это произошло некоторое время назад, когда появился datassette SQuAD.

Пример. Пары для обучения из датасета SQuAD.

In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under gravity. The main forms of precipitation include drizzle, rain, sleet, snow, graupel and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals within a cloud. Short, intense periods of rain in scattered locations are called “showers”.

- What causes precipitation to fall?
    - gravity
- What is another main form of precipitation besides drizzle, rain, snow, sleet and hail?
    - graupel
- Where do water droplets collide with ice crystals to form precipitation?
    - within a cloud

Для каждого вопроса выделен ответ в тексте. SQWAD несколько упростил задачу — нужно найти ответ внутри заданного текста, то есть выделить подстроку внутри строки, которая является ответом.

Чтобы выделить подстроку, необходимо предсказать начало и конец. Это задача классификации. Достаточно для каждого токена в предложении сказать, является ли он началом и концом. Все, что находится между началом и концом, — наш ответ.

SQuAD перевел задачу из Question Answering в общем смысле, где сложно сгенерировать ответ на поставленный вопрос, в задачу поиска ответов в нужном контексте. Это и есть Context-Based Question Answering. При этом SQuAD разметил много данных.

Этот датасет подстегнул интерес людей к построению вопросно-ответных систем: ImagNet, параллельные корпусы текста и др.

Технические аспекты SQuAD

Полезно понять, чем именно SQuAD отличается от других существующих датасетов.

- Авторы предполагали, что люди могут отвечать на вопросы по-разному, поэтому они собирали 3 авторских ответа вместо одного.
- Система оценивала ответы следующим образом:
    - если один из трех авторских ответов был угадан, то системе сразу давался полный балл;
    - использовались различные метрики: precision, recall, гармоническое среднее F1 и т. д.
- Чтобы понять, насколько хороша система, ее сравнивали с человеком. В этом была некоторая проблема, потому что для каждого вопроса было дано 3 ответа от человека. Но для каждого человека было только два других референсных ответа, поэтому люди были в несколько в проигрышном положении относительно машин. Качество людей относительно машины несколько занижалось.
- Игнорировались различные междометия, союзы, пунктуация и т. д.

Это позволяло качественно оценить находимые ответы.

Посмотрим на качество человека в конце 2016 года:

		EM	F1
11	Fine-Grained Gating Carnegie Mellon University (Yang et al. '16)	62.5	73.3
12	Dynamic Chunk Reader IBM (Yu & Zhang et al. '16)	62.5	71.0
13	Match-LSTM with Ans-Ptr (Boundary) Singapore Management University (Wang & Jiang '16)	60.5	70.7
14	Match-LSTM with Ans-Ptr (Sequence) Singapore Management University (Wang & Jiang '16)	54.5	67.7
15	Logistic Regression Baseline Stanford University (Rajpurkar et al. '16)	40.4	51.0
Will your model outperform humans on the QA task?			
	Human Performance Stanford University (Rajpurkar et al. '16)	82.3	91.2

EM — это Exact Match. Всего было примерно 100 тыс. вопросов. Вот что происходило в мае 2020 года:

Rank	Model	EM	F1
	Human Performance *Stanford University* (Rajpurkar et al. '16)	82.304	91.221
1 Apr 10, 2020	LUKE (single model) *Studio Ousia & NAIST & RIKEN AIP*	**90.202**	**95.379**
2 May 21, 2019	XLNet (single model) *Google Brain & CMU*	89.898	95.080
3 Dec 11, 2019	XLNET-123++ (single model) *MST/EOI* http://tia.today	89.856	94.903
3 Aug 11, 2019	XLNET-123 (single model) *MST/EOI*	89.646	94.930
4 Sep 25, 2019	BERTSP (single model) *NEUKG* http://www.techkg.cn/	88.912	94.584
4 Jul 21, 2019	SpanBERT (single model) *FAIR & UW*	88.839	94.635
5 Jul 03, 2019	BERT+WWM+MT (single model) *Xiaoi Research*	88.650	94.393

Машины превзошли человечество в построении ответов на заданные вопросы. Но стоит помнить, что здесь у машин есть некоторое неконкурентное преимущество. У них три референсных ответа, с которыми они сравниваются, а у человека только два.

В дальнейшем SQuAD был доработан, и появился SQuAD 2.0.

2. SQuAD и SberQuAD

Разберемся, в чем отличие SQuAD 2.0 от SQuAD 1.0.

Основной проблемой SQuAD 1.0 было именно то, что для каждого вопроса присутствовал ответ. В реальности далеко не всегда так. Машина не всегда может ответить на вопрос, и было бы хорошо, чтоб она явно говорила: «Я не знаю», а не пыталась найти какую-то подстроку, которая вроде как наиболее хорошо подходит к заданному вопросу.

SQuAD 2.0 предполагал, что в трети вопросов вообще нет ответа, и правильным ответом было No Answer. Это отдельная метка класса для заданного вопроса. Если модель отвечает No Answer, она сразу получает единицу и за EM, и за F1. Если она не отвечает No Answer, то получает ноль.

SQuAD 2.0 показал, что сбор данных — тоже трудоемкая задача, и надо очень правильно предобрабатывать датасет.

Пример. Вопрос из SQuAD:

Genghis Khan united the Mongol and Turkic tribes of the steppes and became Great Khan in 1206. He and his successors expanded the Mongol empire across Asia. Under the reign of Genghis' third son, Ögedei Khan, the Mongols destroyed the weakened Jin dynasty in 1234, conquering most of northern China. Ögedei offered his nephew Kublai a position in Xingzhou, Hebei. Kublai was unable to read Chinese but had several Han Chinese teachers attached to him since his early years by his mother Sorghaghtani. He sought the counsel of Chinese Buddhist and Confucian advisers. Möngke Khan succeeded Ögedei's son, Güyük, as Great Khan in 1251. He

When did Genghis Khan kill Great Khan?

Gold Answers: <No Answer>

Prediction: 1234 [from Microsoft nlnet]

Мы видим историю про Монголию, которая была объединена. Вопрос: «Когда был убит великий хан?» Правильный ответ: «Нет ответа», потому что никаких ответов нет. Речь вообще идем о другом — история про то, как была ослаблена династия Джин в 1234 году, а модель решила, что 1234 — это правильный ответ.

В октябре 2020 года SQuAD 2.0 показывал любопытные результаты:

Rank	Model	EM	F1
	Human Performance *Stanford University* (Rajpurkar & Jia et al. '18)	86.831	89.452
1 Apr 06, 2020	SA-Net on Albert (ensemble) *QIANXIN*	**90.724**	**93.011**
2 May 05, 2020	SA-Net-V2 (ensemble) *QIANXIN*	90.679	92.948
2 Apr 05, 2020	Retro-Reader (ensemble) *Shanghai Jiao Tong University* http://arxiv.org/abs/2001.09694	90.578	92.978
3 Jul 31, 2020	ATRLP+PV (ensemble) *Hithink RoyalFlush*	90.442	92.877
3 May 04, 2020	ELECTRA+ALBERT+EntitySpanFocus (ensemble) *SRCB_DML*	90.442	92.839
4 Jun 21, 2020	ELECTRA+ALBERT+EntitySpanFocus (ensemble) *SRCB_DML*	90.420	92.799
4 Sep 11, 2020	EntitySpanFocus+AT (ensemble) *RICOH_SRCB_DML*	90.454	92.748
4 Mar 12, 2020	ALBERT + DAAF + Verifier (ensemble) *PINGAN Omni-Sinitic*	90.386	92.777
5 Jan 10, 2020	Retro-Reader on ALBERT (ensemble) *Shanghai Jiao Tong University* http://arxiv.org/abs/2001.09694	90.115	92.580
5 Sep 27, 2020	electra+nlayers (ensemble) *oppo.tensorlab*	90.126	92.535

Человечество все равно немножко проиграло даже без нечестного неконкурентного преимущества машин.

Если вы внимательно посмотрите на те модели, которые присутствуют в этой таблице, вы увидите BERT.

SberQuAD

На русском языке есть датасет SberQuAD.

Пример.

Термин Computer science (Компьютерная наука) появился в 1959 году в научном журнале Communications of the ACM, в котором Луи Фейн (Louis Fein) ратовал за создание Graduate School in Computer Sciences (Высшей школы в области информатики) . . . Усилия Луи Фейна, численного аналитика Джорджа Форсайта и других увенчались успехом: университеты пошли на создание программ, связанных с информатикой, начиная с Университета Пердью в 1962.

● Q11870 Когда впервые был применен термин Computer science (Компьютерная наука)?

● Q28900 Кто впервые использовал этот термин?

● Q30330 Начиная с *каого** учебного заведения стали применяться учебные программы, связанные с информатикой?

*Misspelling is intended

Видим, что структура примерно та же самая, — есть английские слова, русские и опечатки (присутствуют специально как некоторый тип регуляризации).

Регуляризация для текстов тоже есть:

- можно выкидывать некоторые слова;
- можно случайно переставлять пару символов — те же самые опечатки;

- можно добавлять случайное слово в какое-то место, зашумляя данные.

Это уже больше в области NLP, прикладных задач.

Результаты были получены — для русского языка они достаточно хорошие:

Model	SberQuAD		SQuAD	
	EM	F1	EM	F1
simple baseline	0.3	25.0	–	–
ML baseline	3.7	31.5	–	–
BiDAF	51.7	72.2	68.0	77.3
DrQA	54.9	75.0	70.0	79.0
R-Net	58.6	77.8	71.3	79.7
DocQA	59.6	79.5	72.1	81.1
BERT	66.6	84.8	85.1	91.8

Table 7: Model performance on SQuAD and SberQuAD; SQuAD part shows single-model scores on test set taken from respective papers.

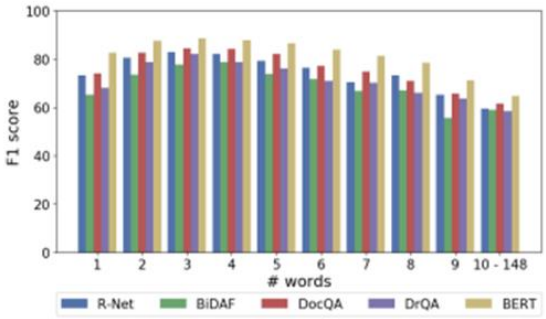


Figure 6: Model performance depending on answer length (# of words).

	% test	R-Net	BiDAF	DocQA	DrQA	BERT
w/ typos	5.7	74.1	66.7	77.5	67.5	81.1
correct	94.3	77.1	72.5	79.6	75.4	85.0
Test set		77.8	72.2	79.5	75.0	84.8

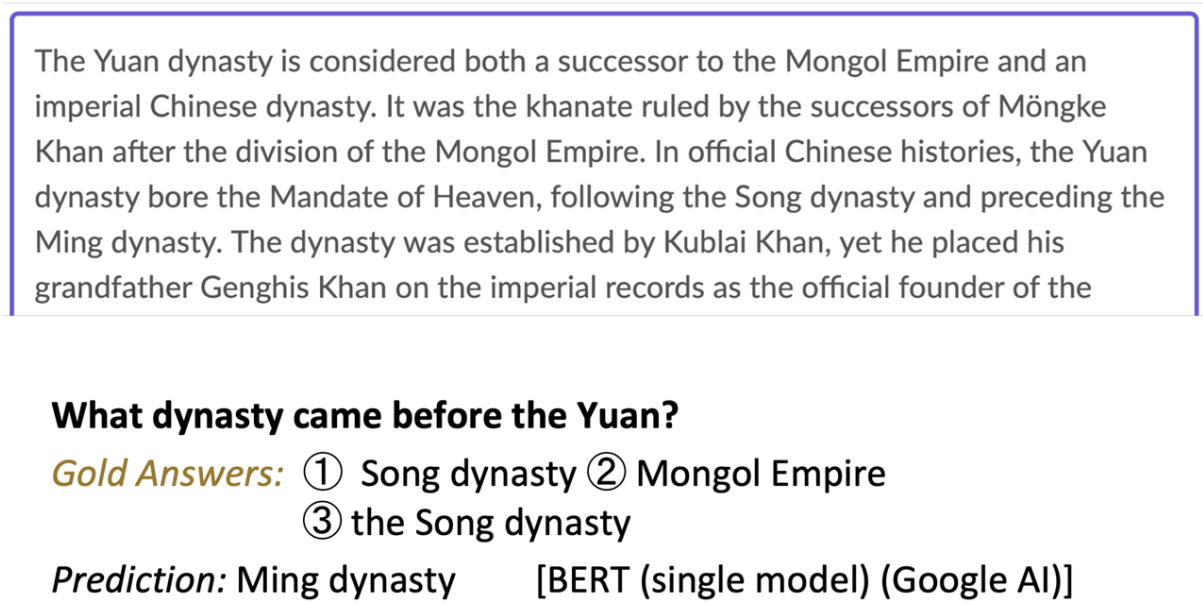
Table 8: Answer quality for misspelled questions.

С точки зрения метрик результаты были чуть пониже по сравнению со SQuAD, потому что датасет SQuAD больше. К тому же для языковых моделей английский язык несколько проще — его проще моделировать, в нем меньше падежей, форм слов и т. д. В 2017 года еще не все ошибки были учтены.

SberQuAD — отличный датасет для русского языка. Это показатель, что для языка необходим хороший датасет, чтобы исследователи этой страны на этом языке сразу же начали глубже погружаться в эту область.

На русском языке возникло гораздо больше решений из области NLP с появлением большого количества открытых наборов данных. Но пока ошибки в вопросно-ответных системах все еще присутствуют.

Пример.



Проблемы SQuAD и SberQuAD

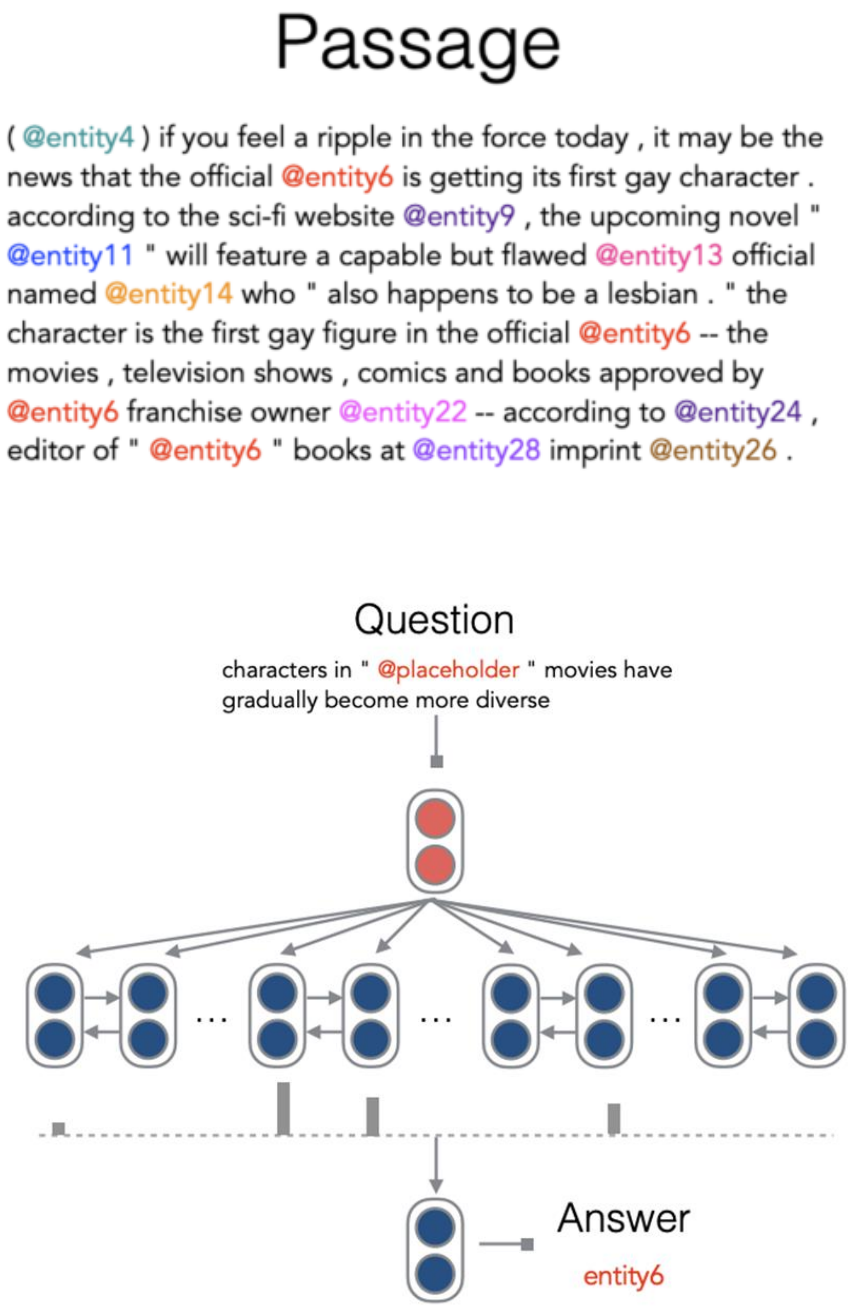
Проблемы всех датасетов SQuAD, SberQuAD, многих популярных языков:

- Невозможно получить систему, которая умеет отвечать да, нет, числом и т. д., потому что ответ — это всегда подстрока. Нет подстроки «Да» или «Нет».
- Нет доступа к внешним данным, то есть все базируется на контексте.

Последнее — это даже не проблема, а свойство. Использование систем, которые базируются на контексте, во многих случаев бывает оправдано.

3. Подходы к построению задачи вопросно-ответных систем

Разработаем несколько моделей, которые помогут нам решать задачу построения вопросно-ответных систем. Рассмотрим следующий вариант возможного решения:



Вопрос — это текст. С текстом работать не очень удобно, если он в виде текста.

Мы уже умеем получать векторное представление для текста. Возьмем какую-нибудь языковую модель, например, BERT или LSTM, прогоним через нее текст и получим вектор, который кодирует наш вопрос. Теперь у нас есть векторное представление вопросов. Нам необходимо найти начало и конец ответа.

Нам не нужно представлять весь контекст в виде одного вектора. Возьмем двунаправленную LSTM или другую рекуррентную сеть. Прогоним наш текст через модель туда-обратно. Теперь для каждой позиции у нас есть некоторый эмбеддинг. Попробуем понять, насколько важен для этого вопроса каждый из этих эмбеддингов (как начало ответа и как конец).

Оценка важности какой-то позиции в предложении для некоторого внешнего состояния — это типичный механизм attention. Мы берем две attention «головы», одна из которых отвечает за начало ответа, другая — за конец, и считаем attention между состояниями для каждой позиции в контексте и скрытым вектором для нашего вопроса.

$$\begin{aligned} \alpha_i &= \operatorname{softmax}_i \mathbf{q}^\intercal \mathbf{W}_s \tilde{\mathbf{p}}_i \\ \mathbf{o} &= \sum_i \alpha_i \tilde{\mathbf{p}}_i \end{aligned}$$

Этот подход неплохо работает и показывает результаты. Две LSTM, одна из которых дает скрытое состояние для каждой позиции в тексте, а вторая дает эмбеддинг — некоторое представление всего вопроса.

Мы можем сделать этот метод лучше — добавим дополнительные данные о тексте:

- **Эмбеддинги на уровне символов.** Если есть эмбеддинги слов, может потеряться информация, например, с большой буквы или с маленькой начиналось слово.

  Пример. Вопрос: «Кто сегодня читал лекцию?» На вопрос «Кто?» обычно отвечают именем собственным, титулом и т. д.

  Эмбеддинги на уровне символов могут помочь нам лучше понять форму слова, его контекст.

- **Лингвистические признаки.**

Пример. Part of Speech (POS) — метки, которые показывают, к какой части речи относится то или иное слово.

- **NER** — метки имен собственных.

Рассмотрим пример POS tagging:

Pred. Tag	Actual Tag	Correct?	Token
PUNCT	PUNCT	✔	[
DET	DET	✔	this
NOUN	NOUN	✔	killing
ADP	ADP	✔	of
DET	DET	✔	a
ADJ	ADJ	✔	respected
NOUN	NOUN	✔	cleric
AUX	AUX	✔	will
AUX	AUX	✔	be
VERB	VERB	✔	causing
PRON	PRON	✔	us
NOUN	NOUN	✔	trouble
ADP	ADP	✔	for
NOUN	NOUN	✔	years
PART	PART	✔	to
VERB	VERB	✔	come
PUNCT	PUNCT	✔	.
PUNCT	PUNCT	✔	]

POS tagging — это одна из подзадач в NLP, которая похожа на задачу языкового моделирования. Но она проще, потому что нам нужно предсказать не следующее слово, а часть речи каждого слова в тексте.

Подходы для решения POS tagging:

- системы, основанные на правилах;
- динамическое программирование;
- CRN (Conditional Random Fields);
- марковские цепи;

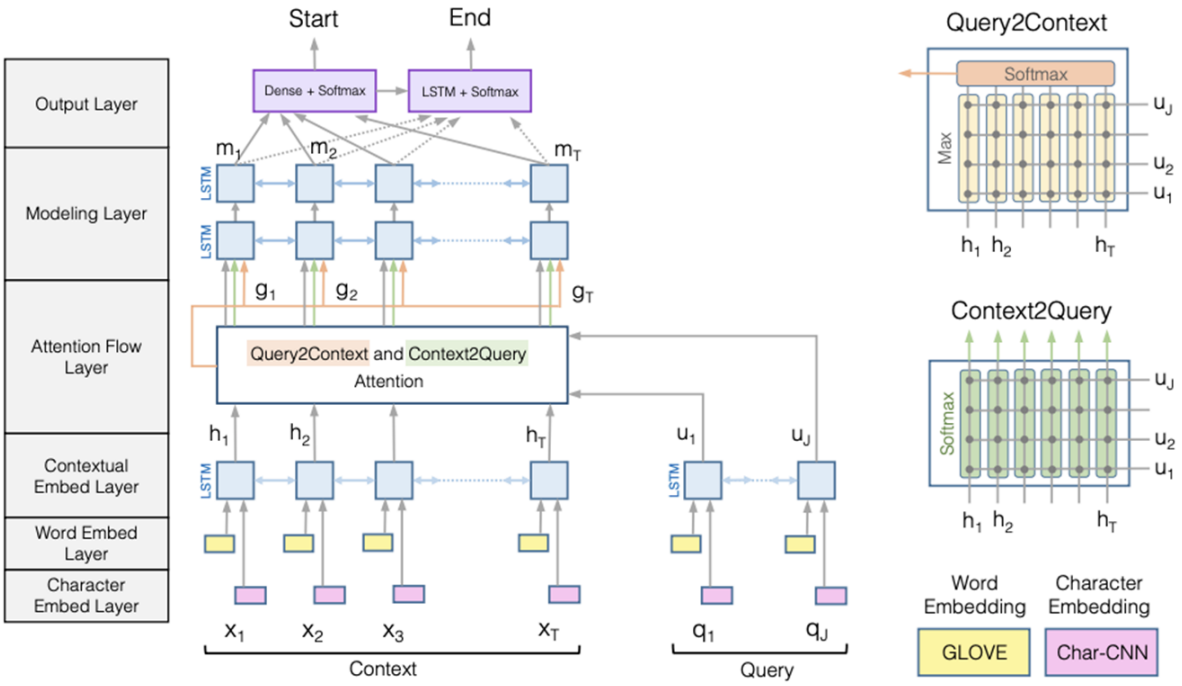
- LSTM, которая будет предсказывать на каждом шаге не следующее слово, а классифицировать символ.

Помимо каких-то признаков, которые дополнительно позволяют понять, что в тексте происходит, мы можем использовать более подходящую архитектуру.

Генерация эмбеддинга для вопроса, игнорируя наличествующий текст для ответа, и генерация состояния для потенциальных ответов, эмбеддингов для всех позиций, игнорируя вопрос, — это не очень хорошая идея. В зависимости от вопроса мы будем по-разному понимать ответ — можно по-разному обращать внимание на те или иные вещи.

BiDAF

Используем механизм внимания. Обратимся к одной из наиболее хороших моделей — BiDAF (Bi-Directional Attention Flow):



Пусть у нас есть ответ. Это контекст, в котором где-то есть ответ или его нет. Также у нас есть вопрос. Он важен для понимания, куда смотреть в ответе. Также важен

контекст, чтобы правильно понять вопрос. А, значит, и контекст влияет на наше восприятие вопроса, и вопрос влияет на восприятие контекста. Оценивать влияние друг на друга мы можем при помощи attention.

В модели BiDAF attention направлен и от вопроса к контексту, и от контекста к вопросу. Для каждой позиции в тексте из контекста и для каждой позиции из вопроса (query) — эмбеддинг из GLoVe (Global Vectors, аналог word2vec).

Еще в модели есть Char- CNN, то есть эмбеддинг с уровня символов, чтобы лучше понимать форму слова. Эмбеддинги конкатенируются, после чего прогоняется LSTM, которая позволяет построить контекстные эмбеддинги для вопроса по отдельности.

Далее считается попарно attention context to query и query to context. После этого мы используем attention score и берем еще один слой, который объединяет все между собой.

Получаем два классификатора, один из которых говорит: «Это начало / Не начало», второй: «Конец / Не конец». Основная идея BiDAF — это attention flow layer.

$$\boldsymbol{S}_{ij} = \boldsymbol{w}_{\text{sim}}^T [\boldsymbol{c}_i; \boldsymbol{q}_j; \boldsymbol{c}_i \circ \boldsymbol{q}_j] \in \mathbb{R}$$

$$\alpha^i = \text{softmax}(\boldsymbol{S}_{i,:}) \in \mathbb{R}^M \quad \forall i \in \{1, \ldots, N\}$$

$$\boldsymbol{a}_i = \sum_{j=1}^{M} \alpha_j^i \boldsymbol{q}_j \in \mathbb{R}^{2h} \quad \forall i \in \{1, \ldots, N\}$$

Считается близость каждой позиции в вопросе к каждой позиции в контексте и наоборот. Затем считается context to query attention и query to context, которые между собой позволяют наладить связи.

$$\boldsymbol{m}_i = \max_j \boldsymbol{S}_{ij} \in \mathbb{R} \quad \forall i \in \{1, \ldots, N\}$$

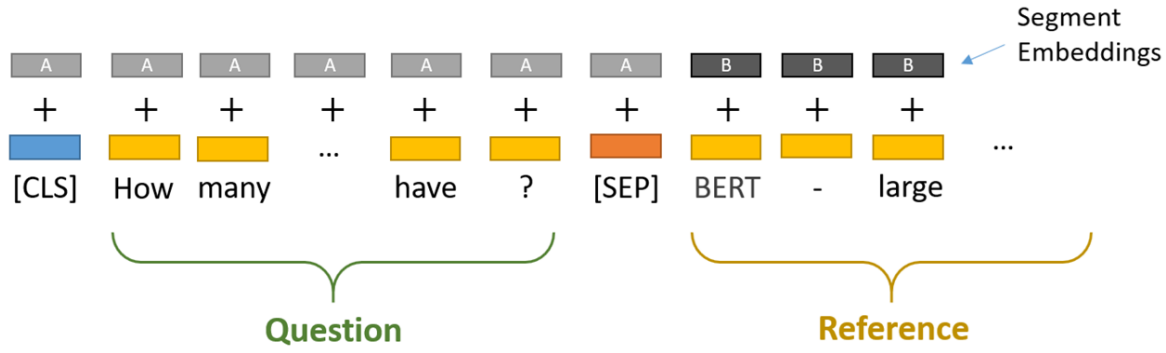
$$\beta = \text{softmax}(\boldsymbol{m}) \in \mathbb{R}^N$$

$$\boldsymbol{c}' = \sum_{i=1}^{N} \beta_i \boldsymbol{c}_i \in \mathbb{R}^2$$

$$\boldsymbol{b}_i = [\boldsymbol{c}_i; \boldsymbol{a}_i; \boldsymbol{c}_i \circ \boldsymbol{a}_i; \boldsymbol{c}_i \circ \boldsymbol{c}'] \in \mathbb{R}^{8h} \quad \forall i \in \{1, \ldots, N\}$$

BERT

Для задачи построения вопросно-ответных систем BERT работает идеально.

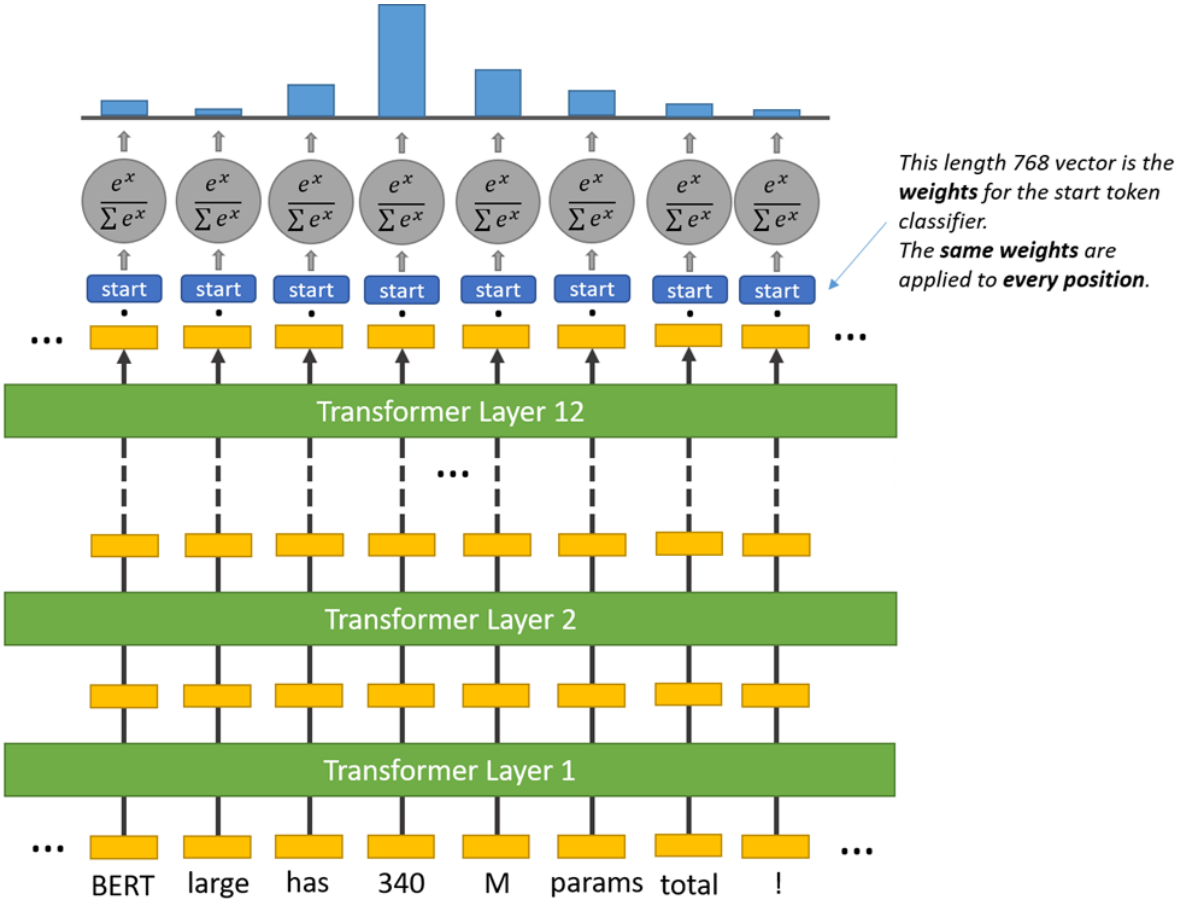


Question: How many parameters does BERT-large have?

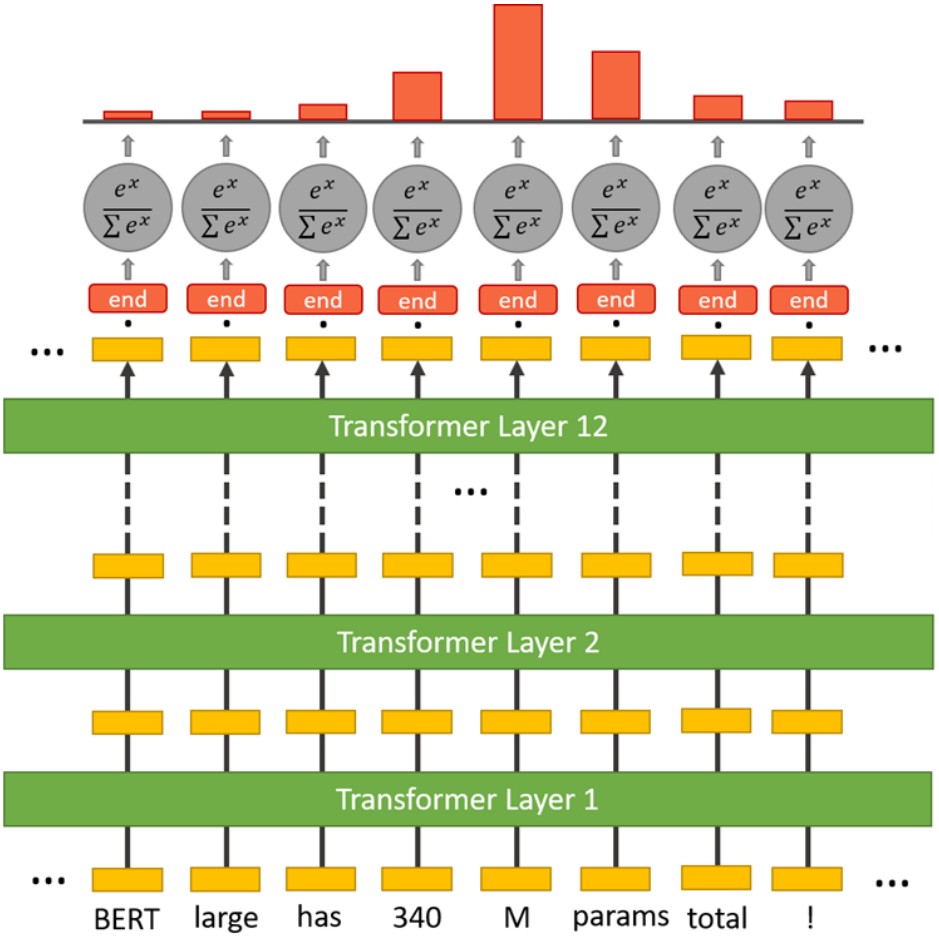
Reference Text: BERT-large is really big... it has 24 layers and an embedding size of 1,024, for a total of 340M parameters! Altogether it is 1.34GB, so expect it to take a couple minutes to download to your Colab instance.

BERT решает задачу классификации каждой позиции в тексте.

У нас есть question & reference — два куска текста, и мы их соединяем друг с другом через сепаратор. Теперь есть вопрос и ответ. Self-attention по умолчанию посчитает attention из question в reference, из reference в question. После этого путем рекомбинаций всех эмбеддингов мы получим какие-то эмбеддинги для каждой позиции, на основании которых мы призовем бинарный классификатор и скажем: «Это старт или не старт?»



Мы видим, что распределение вероятности старта где-то выходит на пик. Далее делаем то же самое, используя те же финальные представления, но только берем другой классификатор и говорим: «Это конец или не конец?»



Получаем, что ответ у нас длится от 340 до params.

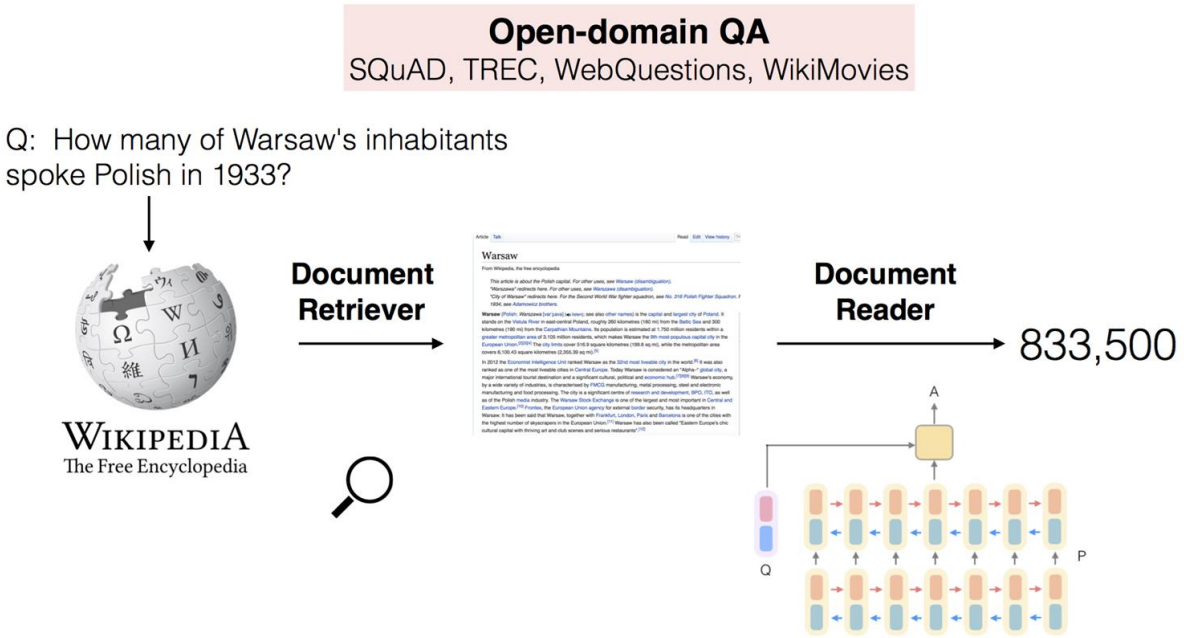
4. Вопросно-ответная система для открытого контекста

Теперь поговорим, как построить вопросно-ответную систему, когда контекст открытый, то есть у нас нет какого-то конкретного домена, в котором мы ищем ответы.

Например, у нас вопрос общей тематики: «Сколько лет было Пифагору на момент написания его первой работы?» Нам необходимо решить, мы будем искать где-то

кусок текста, в котором может быть ответ, или пытаться ответить сразу, если в голове есть ответ.

Посмотрим на вариант с поиском ответа в виртуальной библиотеке:



Например, есть Open-Domain QA, которая работает следующим образом:

- Есть вопрос, на основании которого строится некоторое векторное представление.
- Ведется поиск наиболее подходящих статей на википедии (Википедия, потому что это один из наиболее часто используемых и широких наборов знаний, которые есть у человечества).
- Находятся наиболее потенциально значимые документы, с каждым из которых работаем предыдущим подходом, то есть ищем наиболее подходящий абзац.

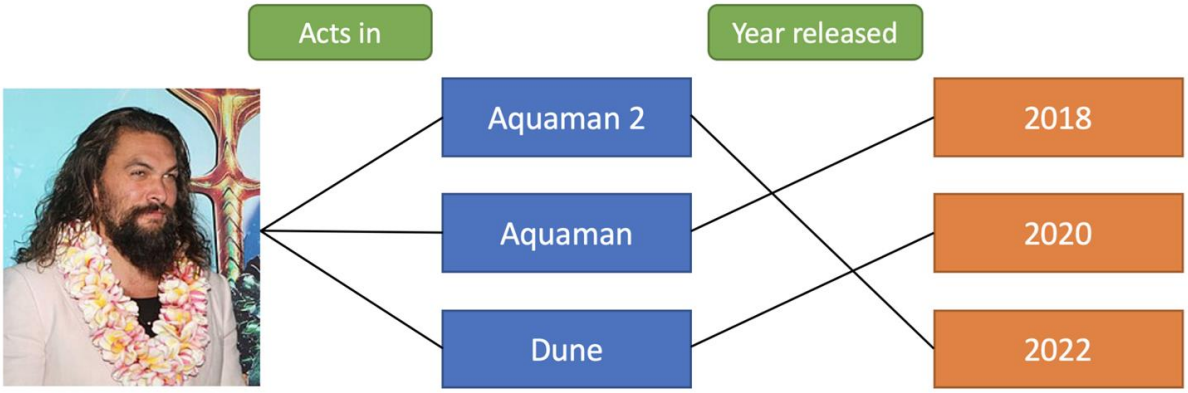
- В наиболее подходящем абзаце ищем необходимый нам ответ.
- Все ответы ранжируются по важности, по правдоподобию, и выдается наиболее правдоподобный ответ или несколько наиболее правдоподобных ответов с метками.

Этот пайплайн многоэтапный, поэтому вероятность отказа гораздо выше. Но прелесть в том, что даже такой пайплайн вы можете запустить на мощной машине.

Но есть и гораздо более сложные задачи.

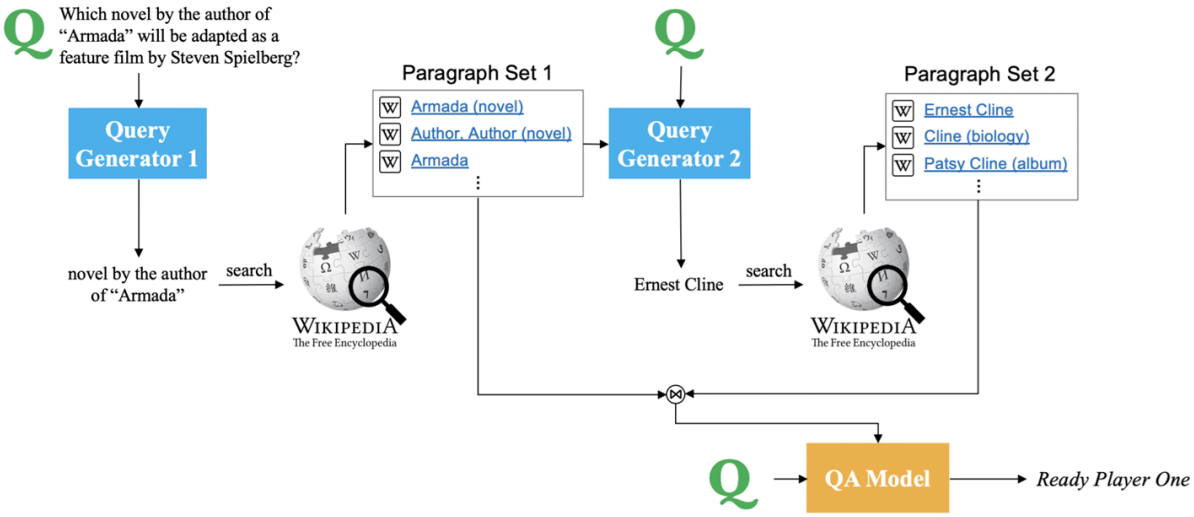
Пример. Ситуация, где нужно разрешить сначала внутреннюю неопределенность, а потом уже отвечать на вопрос.

Example question: “What is the Aquaman actor’s next movie?”



Внутри вопроса содержится еще один вопрос: «Кто такой актер Аквамена?» А потом: «В каком следующем фильме он будет играть?» Для начала нужно понять, что такое «Аквамен», а также что есть человек, который его играет. Когда мы это поняли, нужно найти следующий фильм для этого актера.

Каскадная ситуация, поэтому необходимо решать ее каскадным образом. Для этого есть несколько подходов. Например, есть любопытный подход, который рекурсивно бьет вопрос на подвопросы, отвечает на подвопросы. Как только они получили разрешение, возвращается назад и делает это вновь и вновь.



Это вся та же модель с Open-Domain Question Answering, но теперь она повторяет сама себя рекурсивно до некоторой глубины.

Эти системы не столь устойчивы, но при этом они работают. И, конечно, здесь огромный простор для дальнейших исследований и маневра.

Теперь поговорим, почему системы, которые умеют отвечать на вопросы на основании контекста (Context-Based), могут быть предпочтительнее, чем Open-Domain Question Answering. То есть почему, используя закрытый контекст, мы можем получить решение лучше, чем используя открытый контекст или доставая ответы на вопросы из головы.

Вспомним одно очень важное свойство всех языковых моделей, которые самостоятельно генерируют ответы на вопросы. Пусть у нас есть языковая модель. Предположим, она сама генерирует ответ на вопрос — не смотрит ни в какие базы данных. Или языковая модель генерирует какой-то текст. Проблема такой генерации в том, что мы ее не контролируем.

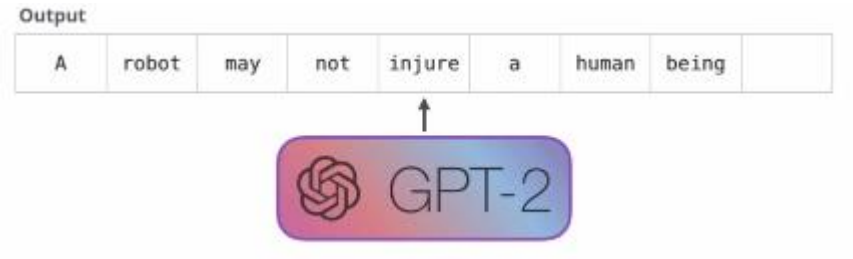
Мы гораздо хуже контролируем нейромашинный перевод, потому что модель сама что-то думает, генерирует какие-то последовательности. Мы можем еще как-то посмотреть на attention, куда она смотрела, но нам сложно сказать, как ее миллионы параметров влияют на те или иные предсказания.

Когда тема чувствительная, например, юридические или медицинские вопросы, мы не хотим давать ответ из головы нашей модели. Нам лучше дать какую-то медицинскую энциклопедию, в которой точно все правильно, и разрешить модели подглядывать только в нее и искать там правильный ответ. Тогда гарантированно ответом будет некоторая подстрока уже провалидированных данных.

Еще один пример, почему такие вопросно-ответные системы часто используют в качестве ботов-помощников в различных приложениях на телефонах или сайтах — если бот внезапно оскорбит человека или скажет что-то неуместное, можно заплатить большой штраф. Поэтому во многих «чувствительных» областях, особенно в прикладных решениях, предпочитают больше контроля, но меньше качества. Поэтому часто Context-Based Question Answering используется вместе с внутренним набором данных, внутренней базой знаний, из которой берется информация.

5. GPT-2 и GPT-3

GPT — это просто декодер трансформера, который учится генерировать текст шаг за шагом. Ровно так же, как языковые модели на основе тех же самых LSTM, RNN и т. д.



Единственное отличие GPT, особенно GPT-2, — в них много параметров. В GPT-2 1,5 млрд параметров. В остальном это все тот же самый трансформер, из которого взяли декодер, который учится предсказывать следующее слово.

Не все модели удается обучить, потому что они слишком огромные. При этом GPT-2 обучался на 8 миллионах веб-страницах. Результаты были феноменальны. На задачах на работу с языком, вроде генерации языка, Question Answering вне всякого контекста, суммаризации текста (попытки сжать текст), перевода данная модель работала вообще без дообучения.

Пример.

EXAMPLES

Who wrote the book the origin of species?

Correct answer: *Charles Darwin*
Model answer: `Charles Darwin`

What is the largest state in the U.S. by land mass?

Correct answer: *Alaska*
Model answer: `California`

GPT-2 также умеет заниматься языковым моделированием.

EXAMPLE

Both its sun-speckled shade and the cool grass beneath were a welcome respite after the stifling kitchen, and I was glad to relax against the tree's rough, brittle bark and begin my breakfast of buttery, toasted bread and fresh fruit. Even the water was tasty, it was so clean and cold. It almost made up for the lack of...

Correct answer: *coffee*
Model answer: `food`

Модель ответила хорошо, потому что, возможно, из контекста было не совсем достаточно информации.

Пример. Модели на вход пришел текст на французском:

EXAMPLE

French sentence:
Un homme a expliqué que l'opération gratuite qu'il avait subie pour soigner une hernie lui permettrait de travailler à nouveau.

Reference translation:
One man explained that the free hernia surgery he'd received will allow him to work again.

Model translation:
`A man told me that the operation gratuity he had been promised would not allow him to travel.`

Она неплохо его перевела. При этом GPT-2 никто не учил на задаче машинного перевода. Модель видела в огромных количествах только переписки между людьми.

GPT2 — это был лишь вестник того, что происходит. На графике можно посмотреть, как развивались модели, и в том числе по числу параметров.

В ELMo — 94 миллиона параметров. Первый GPT — на 110 миллионов параметров, GPT-2 — полтора миллиарда параметров.

Компания Nvidia выпустила свою модель Megatron–LM на 8,3 миллиарда параметров. На GPU с применением CUDA, технологии ускорения вычислений, в основном нейронные сети и обучаются.

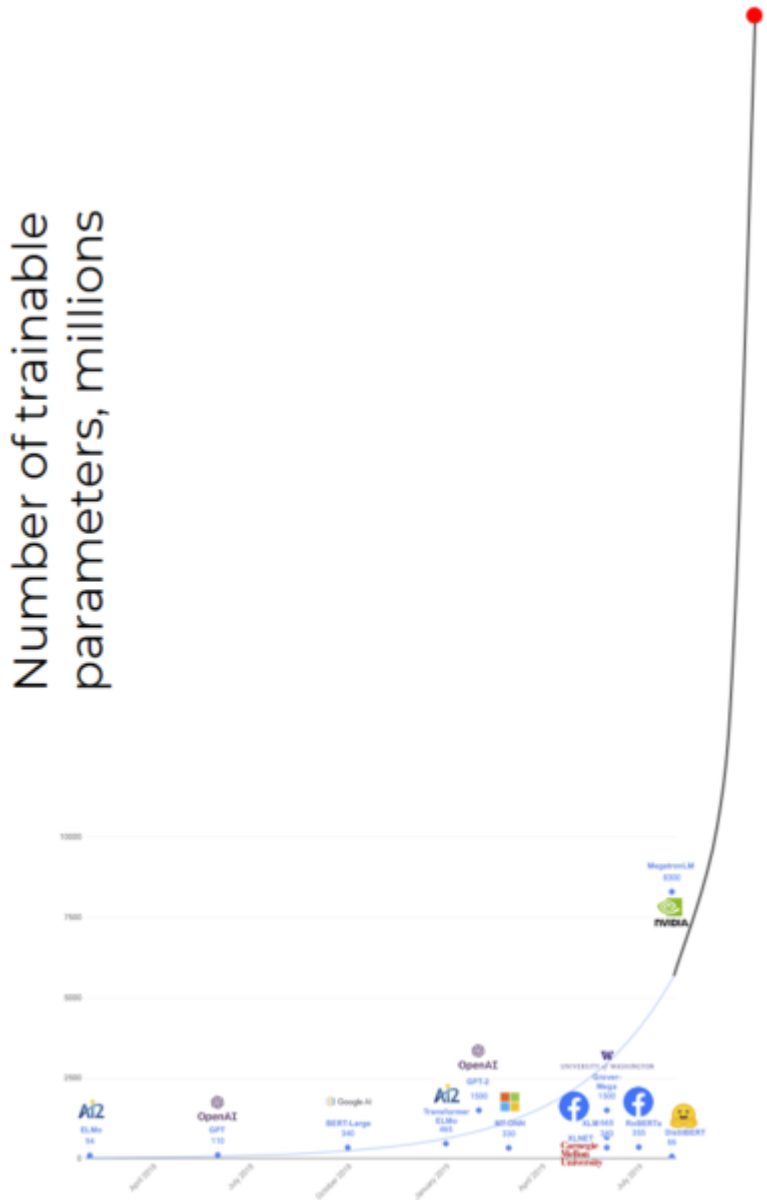
Сейчас движение идет сразу в двух направлениях. Человечество пытается строить более сложные и глубокие модели, чтобы добиться больших результатов. С другой стороны, многие исследователи двигаются в сторону упрощения моделей при сохранении их качеств, потому что не все модели можно запустить на обычном компьютере.

В 2019 году появилась DistillBert, которая обладает лишь, 66 миллионами параметров. При этом она показывает практически такое же качество, что и BERT base, и работает быстрее.

	Nb of parameters (millions)	Inference Time (s)
GLUE BASELINE (ELMo + BiLSTMs)	180	895
BERT base	110	668
DistilBERT	66	410

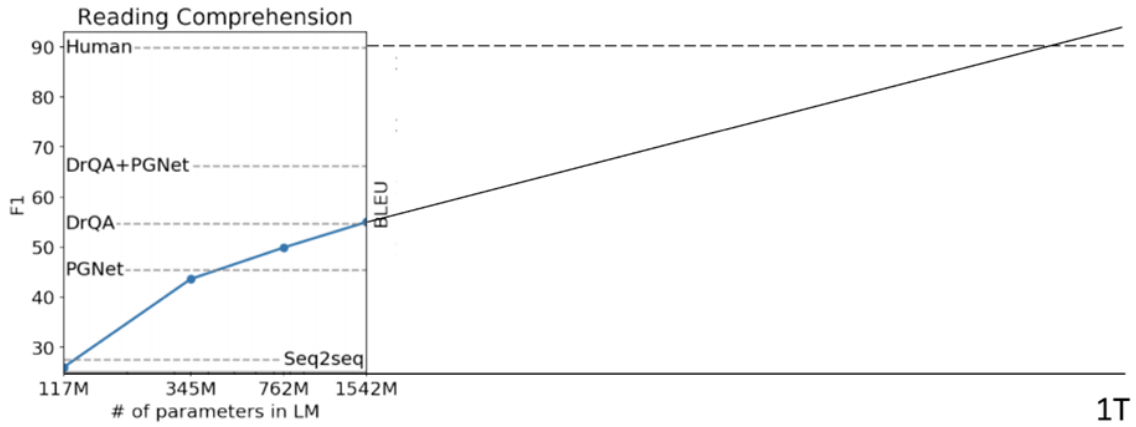
Для его построения авторы использовали процедуру дистилляции, предложенную Джеффри Хинтоном в 2015 году. Она позволяет продистиллировать информацию из большой сложной модели в более маленькую.

График выше — на момент 2019 года. Вот что произошло в 2020 году:



GPT-3 вышел в мае 2020 года. В GPT-3 175 млрд параметров. Сейчас есть модели, в которых количество параметров около триллиона. Обучить такие модели может позволить себе только несколько компаний в мире и правительства крупнейших стран. Чтобы запустить такие модели, должны быть огромные вычислительные мощности.

Приведем пример из лекции Hypothesis from Stanford CS224N Lecture 20 (2019), где показывается экстраполяция того, когда человечество создаст модель, которая будет сопоставима по качеству в задаче понимания текста с самим человеком.



Оценка где-то около триллиона параметров.

С каждым годом появляются все более сложные модели, но с такими моделями нет доступа к вычислительным мощностям. Так вырождается сообщество исследователей, дальнейшее развитие тормозится, происходит балансировка между сложными и простыми моделями.

7. Подробнее о GPT

Модель GPT-3 используют для разных задач, в том числе чтобы она писала код по запросу. Модель может верстать простые сайты по их текстовому описанию. Возможно, в дальнейшем вместо создания датасетов и написания кода можно писать запросы в интерфейсы вроде OpenIA GPT-3.

MiniGPT Андрея Карпатого позволяет понять, как работает GPT непосредственно в коде, и как можно его обучить. MiniGPT — это тот же GPT, который позволяет решать те же задачи, пусть и попроще.

Дополнительные материалы для самостоятельного изучения

1. Natural Language Processing with Deep Learning with Deep Learnig
2. SQuAD: 100,000+ Questions for Machine Comprehension of Text
3. SQuAD website
4. SberQuAD — Russian Reading Comprehension Dataset: Description and Analysis
5. SDSJ 2017 problem B
6. A Thorough Examination of the CNN/Daily Mail Reading Comprehension Task
7. Bidirectional Attention Flow for Machine Comprehension
8. Question Answering with a Fine-Tuned BERT
9. DrQA
10. Answering Complex Open-domain Questions at Scale
11. The Illustrated GPT-2 (Visualizing Transformer Language Models)
12. Smaller, faster, cheaper, lighter: Introducing DistilBERT, a distilled version of BERT
13. Transformer
14. OpenAI Transformer
15. ELMO
16. BERT
17. BERTology
18. GPT
19. GPT-2
20. GPT-3

21. The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning)
22. GPT-3 A Hitchhiker's Guide
23. Andrej Karpathy twitter
24. miniGPT by Andrej Karpathy
25. Transformer-XL
26. BiDAF
27. SQuAD
28. SberQuAD paper
29. GPT-3 trained on Russian texts
30. deeppavlov.ai
31. Natasha project