Глубокие представления слов и документов

Математические методы анализа текстов осень 2019

Попов Артём Сергеевич

МГУ имени М. В. Ломоносова, факультет ВМК, кафедра ММП

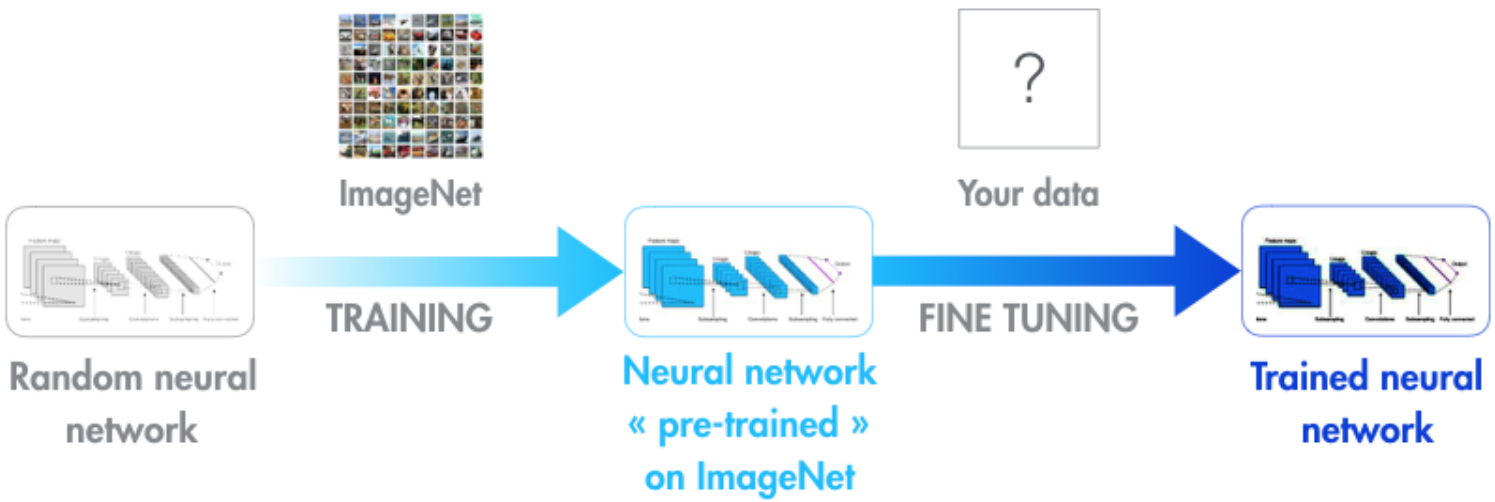
Перенос обучения (transfer learning)

Модель, обученную по большому массиву данных, можно использовать для решения задач на других данных.

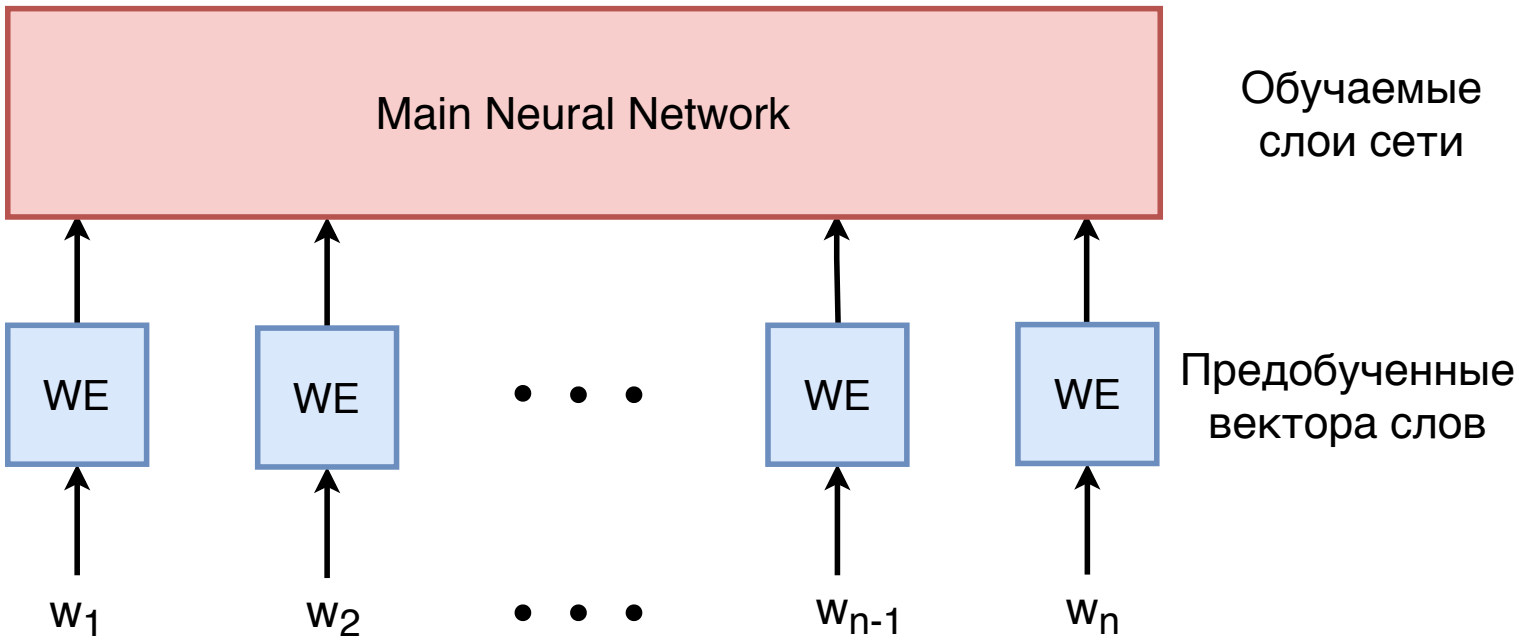
Пример

Датасет ImageNet: 14 млн изображений, 20K категорий

1. Сеть предобучается по ImageNet
2. Сеть дообучается под задачу



Использование векторных представлений слов



Если обучающих данных мало: используем предобученные представления (можно дообучать вместе с основной сетью).

Иначе: тренируем представления и сеть одновременно.

Использование эмбеддингов слов для transfer learning

- Проблема омонимии: одно слово — одно представление.

  Ваша карта заблокирована.
  Я не могу найти на карте ваш офис.

- Невысокая эффективность — не можем предобучить сложные зависимости.
- Привязка только к словам.

Задача переноса обучения (transfer learning) для текстов

Дано: $D = \{d_1, \ldots, d_N\}$ — неразмеченная коллекция
d — один документ, $d = \{w_1, \ldots, w_n\}$
$w \in W$ — слово из словаря коллекции

Необходимо: построить по D модель, которую можно будет использовать при обучении **другой** модели на **другой** обучающей выборке.

Критерий качества: качество решения итоговой задачи при использовании предобученной модели выше чем без её использования.

Виды transfer learning в текстах

По типу использованной информации:

- Языковое моделирование
- Дистрибутивная гипотеза для слов
- Дистрибутивная гипотеза для предложений

В последовательном тексте по i-ому предложению можно восстановить $i-1$ и $i+1$ предложения.

По принципу применения:

- Фиксированные представления для слов (pre-trained representations)
- Перестройка предобученной модели (fine-tuning)

Языковое моделирование (напоминание)

Задача языкового моделирования:
оценить вероятность появления любой последовательности слов $(w_1, \ldots, w_n)$ в «реальном» тексте.

Языковая модель — модель, позволяющая вычислить вероятность $p(w_1, \ldots, w_n)$ для любых $w_1, \ldots, w_n \in W$.

Для языкового моделирования можно использовать нейронные сети, работающие с последовательностями.

Как можно использовать языковое моделирование для transfer learning?

Языковое моделирование (напоминание)

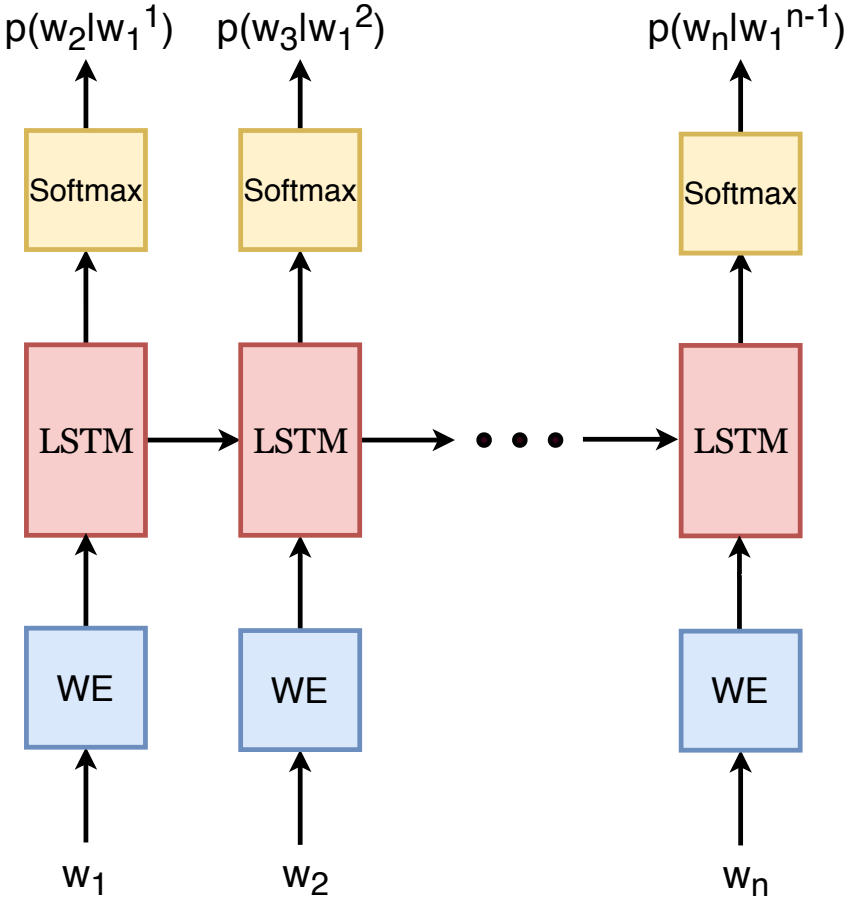
Задача языкового моделирования:
оценить вероятность появления любой последовательности слов $(w_1, \ldots, w_n)$ в «реальном» тексте.

Языковая модель — модель, позволяющая вычислить вероятность $p(w_1, \ldots, w_n)$ для любых $w_1, \ldots, w_n \in W$.

Для языкового моделирования можно использовать нейронные сети, работающие с последовательностями.

Как можно использовать языковое моделирование для transfer learning?
Использовать для представлений скрытые слои модели.

Языковое моделирование для transfer learning



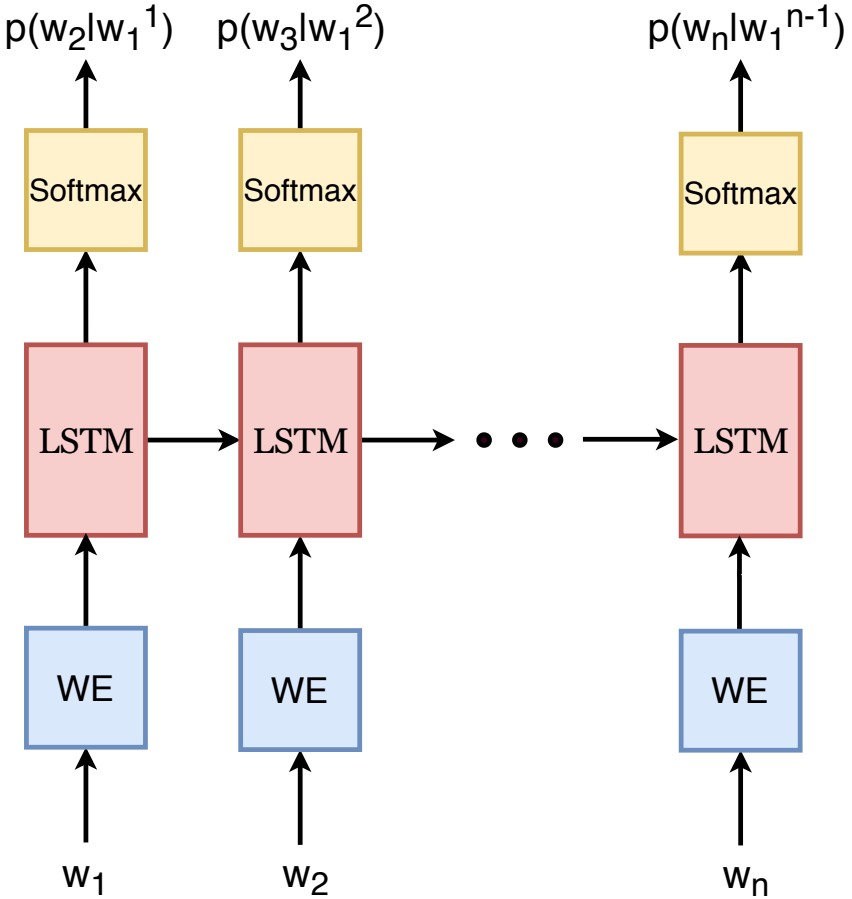
$$v_i = WE(w_i)$$
$$h_i, c_i = LSTM(v_i)$$

Можно использовать в качестве представлений:

- v_i
- h_i
- $\gamma v_i + (1 - \gamma) h_i$
- γ может быть обучаемым

Как добавить двунаправленность в сеть?

Языковое моделирование для transfer learning



$$v_i = WE(w_i)$$
$$h_i, c_i = LSTM(v_i)$$

Можно использовать в качестве представлений:

- v_i
- h_i
- $\gamma v_i + (1 - \gamma) h_i$
- γ может быть обучаемым

Как добавить двунаправленность в сеть?

Использовать конкатенацию двух языковых моделей.

Функционал ELMO (Embeddings from Language Models)

Для документа $d = \{w_1, \ldots, w_n\}$ функционал обучения:

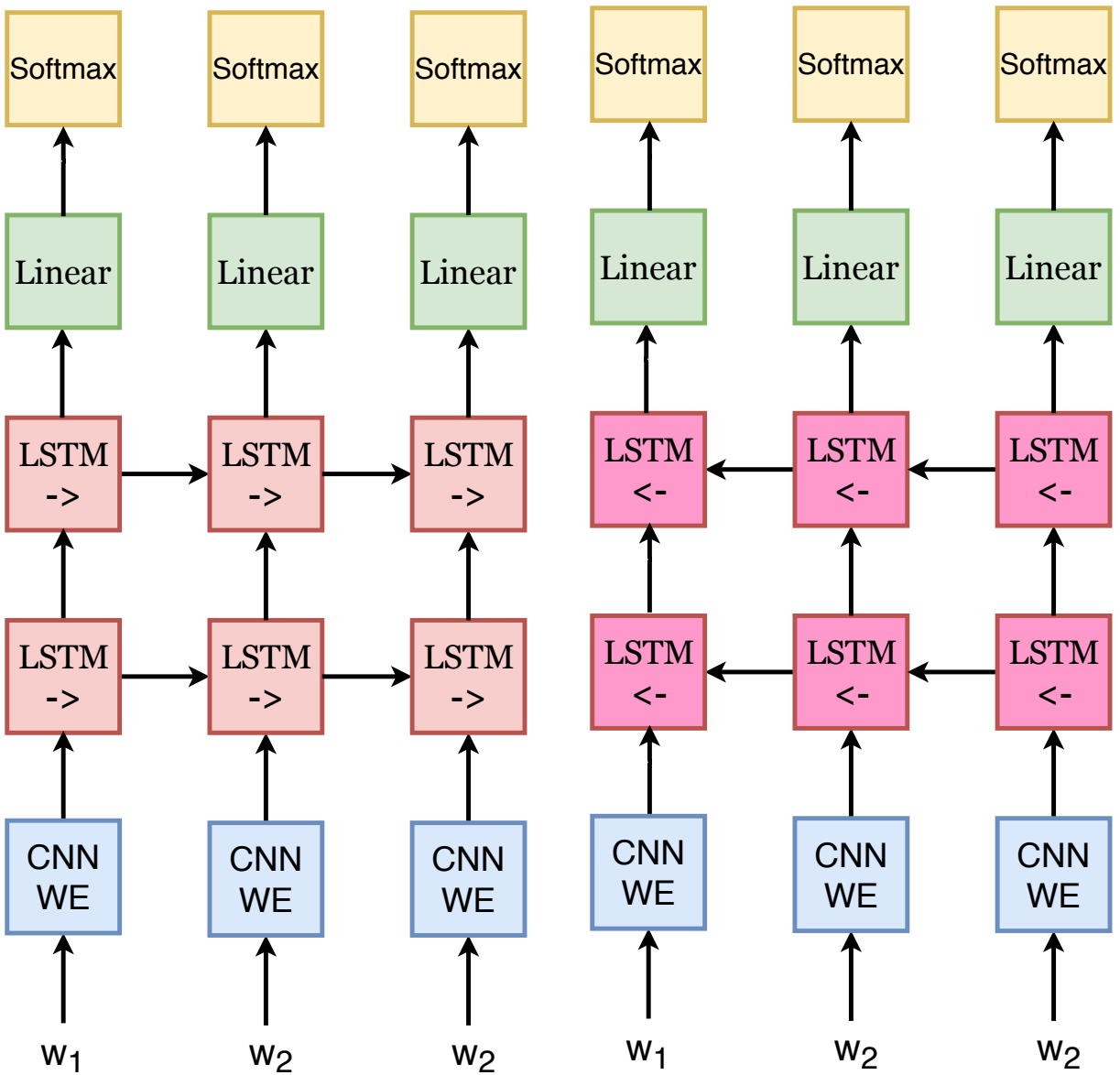
$$\sum_{i=1}^{n} \Big(\log p\big(w_i | w_1^{i-1}, \theta_x, \theta_{\rightarrow}, \theta_s\big) + \\ + \log p\left(w_i | w_{i+1}^{N}, \theta_x, \theta_{\leftarrow}, \theta_s\right)\Big) \rightarrow \max_{\Theta}$$

- θ_x — параметры представлений
- $\theta_{\rightarrow}, \theta_{\leftarrow}$ — параметры каждой из LSTM сетей
- θ_s — параметры выходного линейного слоя
- $\Theta = \{\theta_x, \theta_{\rightarrow}, \theta_{\leftarrow}, \theta_s\}$

Важно. В ELMO выходы двух рекуррентных сетей не конкатенируются перед линейным слоем.

[1] Peters et al (NAACL 2018) Deep contextualized word representations

ELMO



ELMO в деталях

В классической архитектуре:

- Два уровня LSTM ($L = 2$)
- Есть residual связи между разными уровнями LSTM
- CNN char-based представления для слов
- Иногда использует layer нормализацию для лучшего качества решения

ELMO представления

ELMO представления в модели вычисляются по формуле:

$$ELMO_w = \gamma^{task} \sum_{j=0}^{L} s_j^{task} h_k^j,$$

$$\sum_{j=0}^{L} s_j^{task} = 1, \qquad s_j^{task} \geqslant 0 \qquad \forall j.$$

- h_k^j — представления с j-го слоя сети
- s_j^{task} и γ^{task} — дообучаются под конкретную задачу при фиксированных h_k^j

Процесс применения ELMO

Дано:

- задача, входные данные — последовательность слов
- модель, решающая эту задачу

Алгоритм применения ELMO:

1. Конкатенируем $ELMO_w$ представления с представлениями слов входной последовательности
2. Фиксируем представления h_k^j
3. Учим всю исходную архитектуру и веса $s_j^{task}, \gamma^{task}$

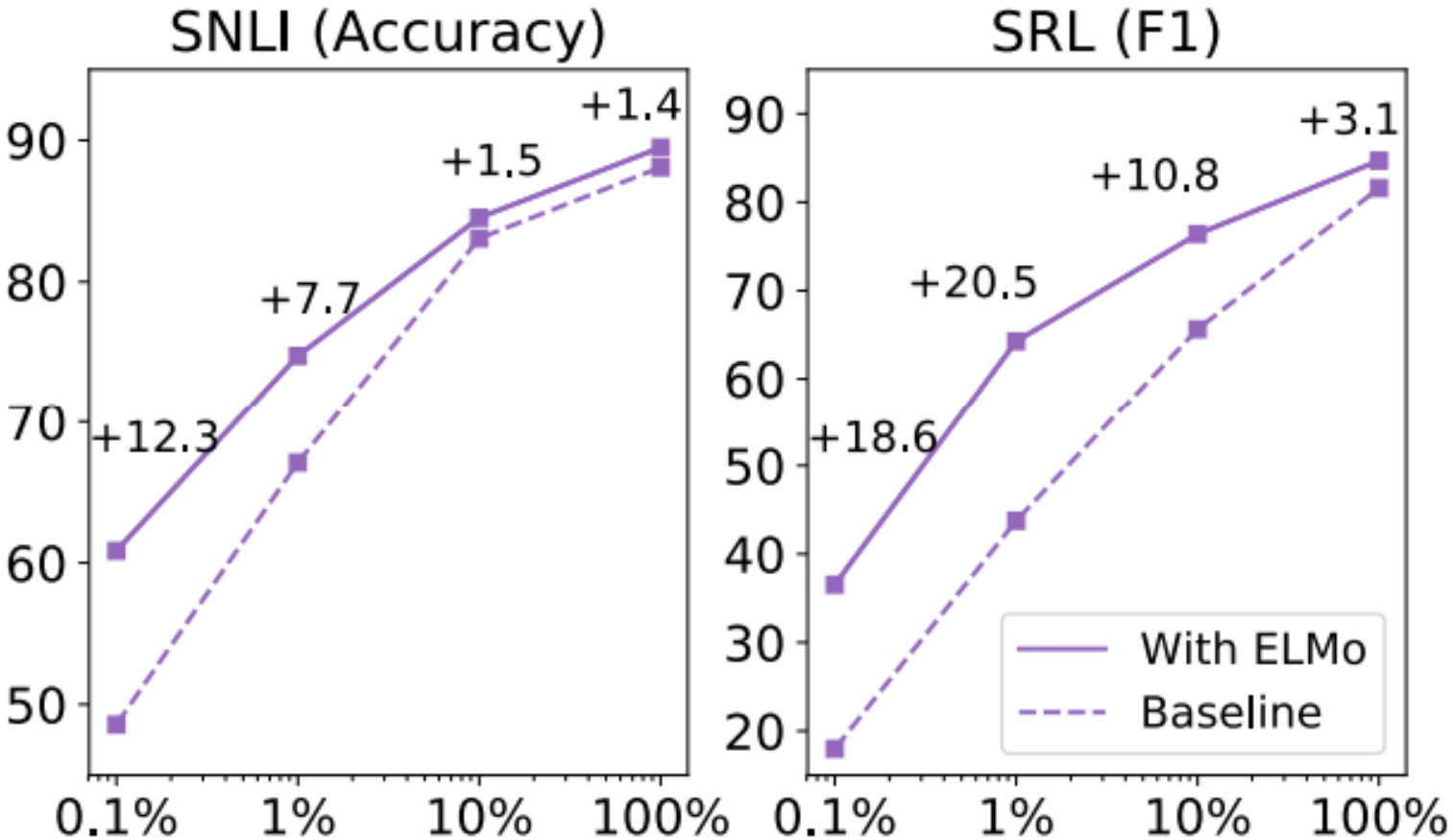
А можно конкатенировать $ELMO_w$ к выходному слою или подавать вместо эмбеддингов...

Результаты ELMO

Повышение качества по всем рассмотренным задачам!

TASK	PREVIOUS SOTA	BASELINE	ELMO + BASELINE
SQuAD	84.4	81.1	85.8
SNLI	88.6	88.0	88.7
SRL	81.7	81.4	84.6
Coref	67.2	67.2	70.4
NER	91.93	90.15	92.22
SST-5	53.7	51.4	54.7

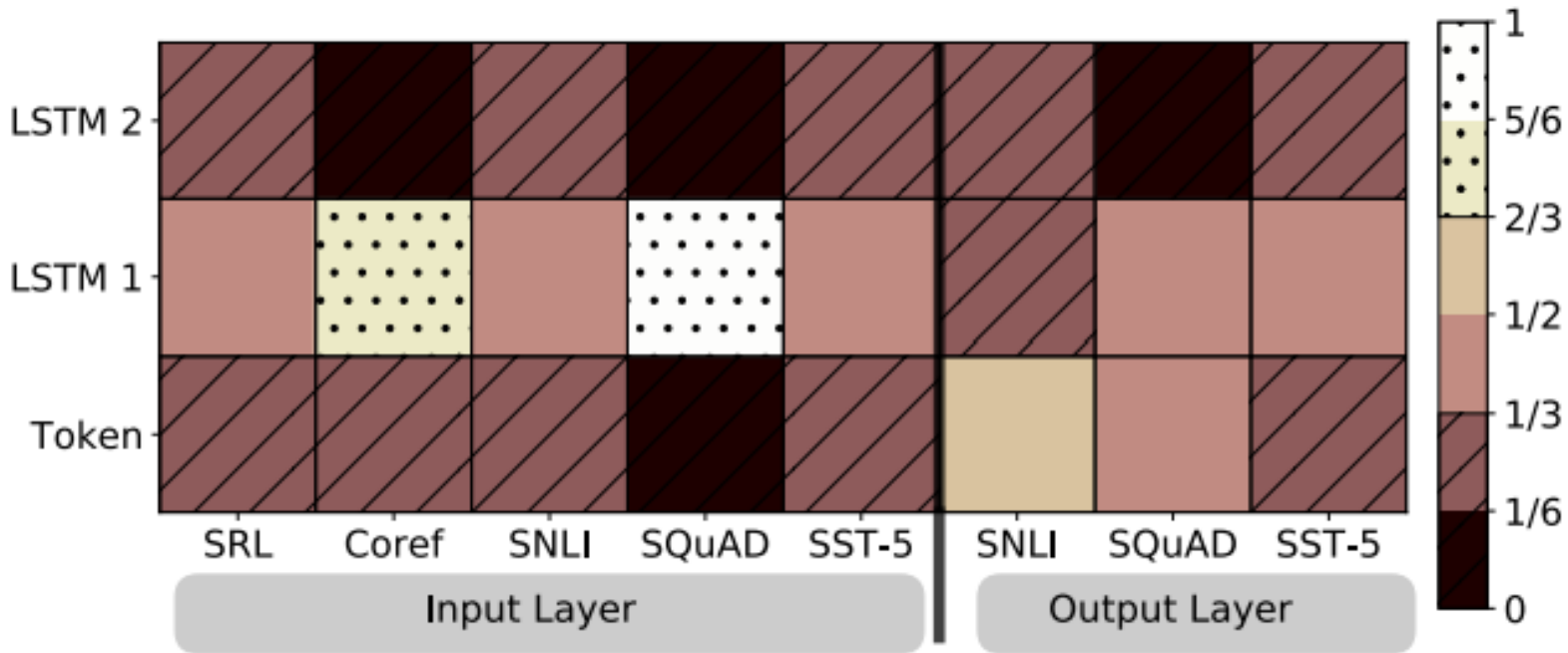
Зависимость качества от размера обучающей выборки



Если данных много, ELMO не даёт большого прироста ...

Значения s_j в зависимости от задачи

По оси y — слои модели, по оси x — задачи:



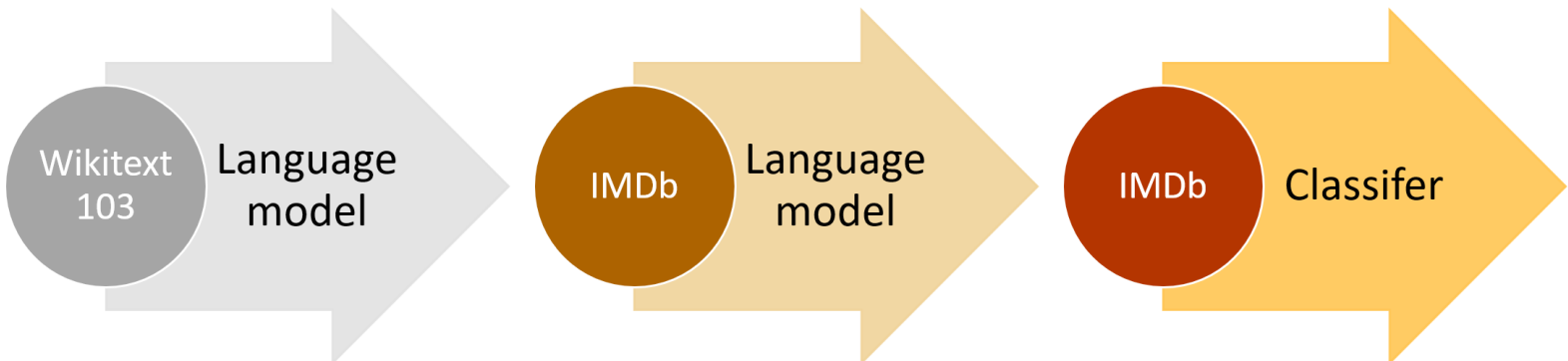
Для разных задач важны разные уровни модели!

Резюме по ELMO

- Мощная модель, позволяющая получить прирост качества в любых задачах
- Использует только рекуррентные сети, поэтому быстро переобучается с увеличением числа слоёв
- По умолчанию дообучаются только коэффициенты взвешивания

ULMFit (Universal Language Model Fine-tuning)

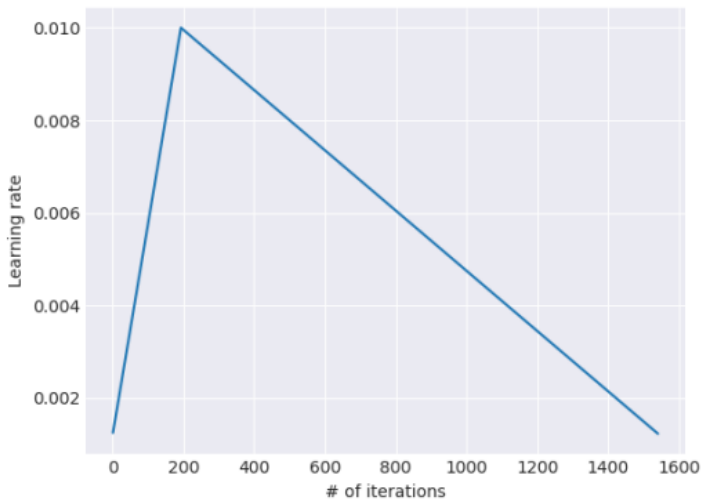
1. Обучаем нейросеть как языковую модель на неразмеченном корпусе
2. Дообучаем нейросеть как языковую модель на обучающем корпусе
3. Добавив дополнительные слои, дообучаем нейросеть на обучающем корпусе

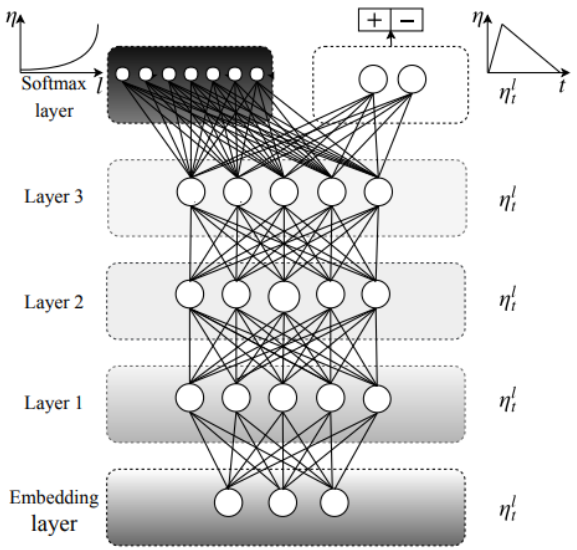


[1] Howard et al (2018) Universal Language Model Fine-tuning for Text Classification

Различные детали обучения ULMFit

- Две однонаправленных сети по разным направлениям
- Три уровня LSTM (L = 3)
- На третьем этапа сначала дообучается последний слой, затем последние два и т.д. (gradual unfreezing)
- Специальная схема выбора learning rate (STLR) для разных слоёв





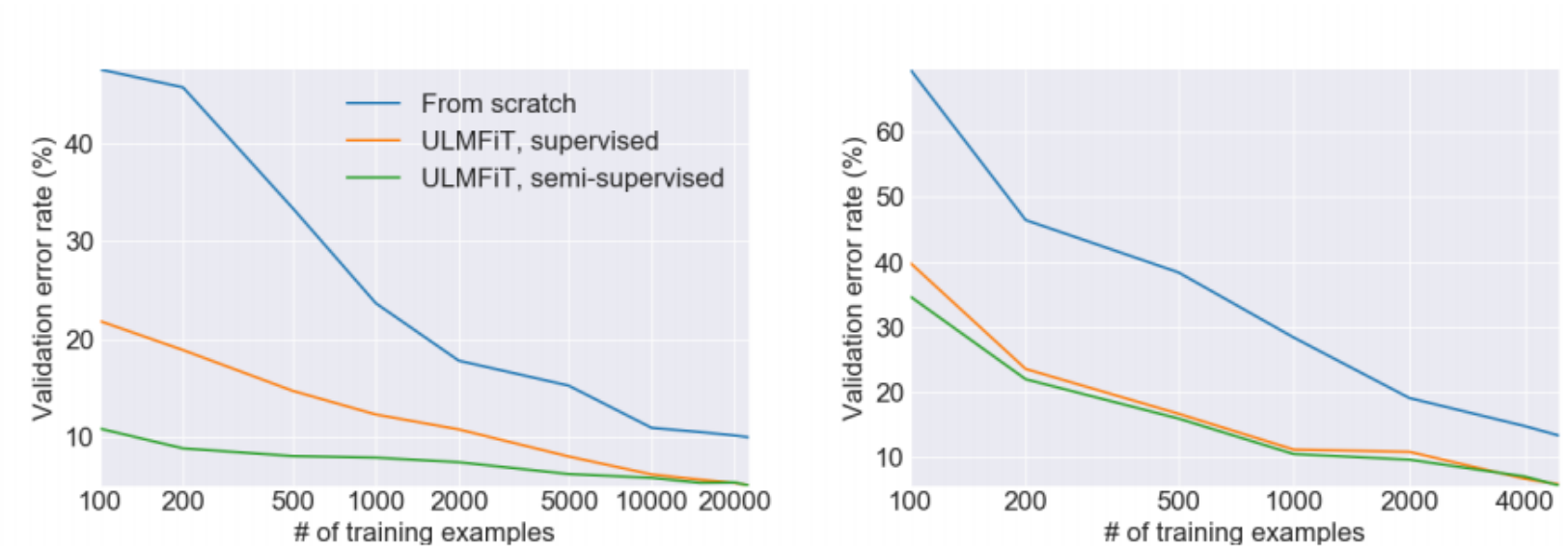
Ошибка классификации ULMFit на разных датасетах

Повышение качества по всем датасетам:

	AG	DBpedia	Yelp-bi	Yelp-full
Char-level CNN (Zhang et al., 2015)	9.51	1.55	4.88	37.95
CNN (Johnson and Zhang, 2016)	6.57	0.84	2.90	32.39
DPCNN (Johnson and Zhang, 2017)	6.87	0.88	2.64	30.58
ULMFiT (ours)	**5.01**	**0.80**	**2.16**	**29.98**

Выигрыш при использовании semi-supervised обучения

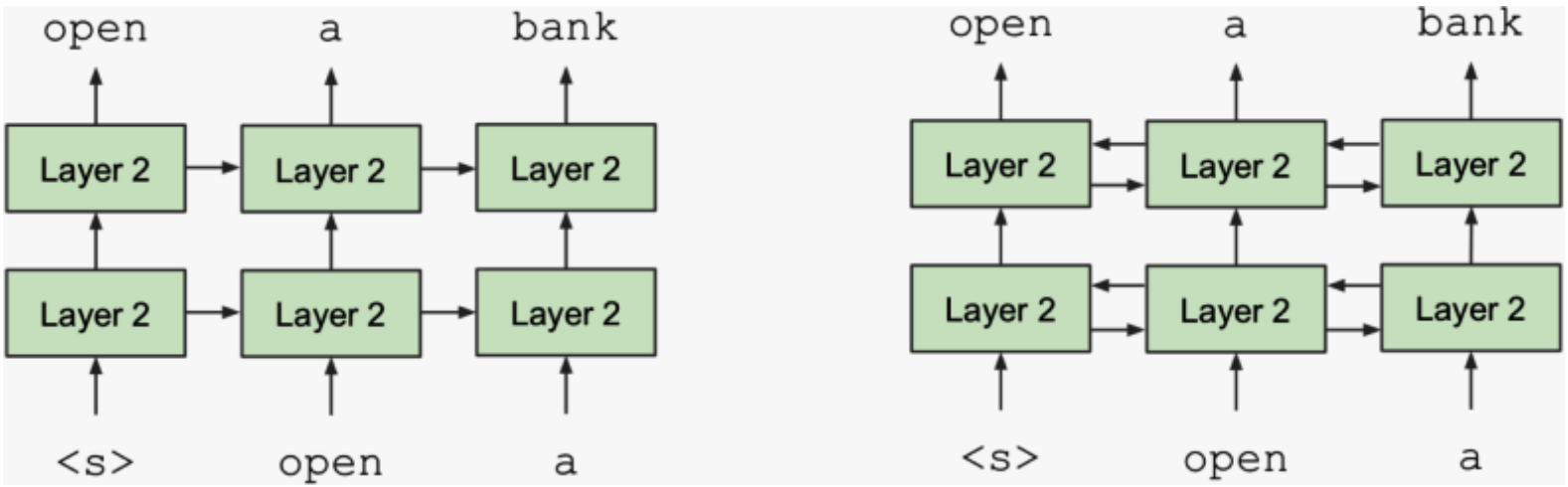
Ошибка на валидации на датасетах IMDb и TREC-6:



Использование semi-supervised подхода может дать преимущество, если обучающих данных не очень много.

Использование трансформеров для базовой модели

Проблема: при обучении языковой модели мы можем использовать либо левый, либо правый контекст.



Как модифицировать процесс обучения, чтобы возможно было использовать трансформер?

Masked-LM

Задача masked language model:

1. Заменяем $k\%$ слов в входной последовательности на специальный токен [MASK]
2. Предсказываем моделью замаскированные токены

Пример:

Next sentence prediction

Задача next sentence prediction:

1. Подаём на вход два предложения, разделённые специальным токеном [SEP]
2. Первый токен — специальный токен [CLS]
3. На выходе от токена [CLS] необходимо предсказать, следует ли одно предложение за другим в тексте

Пример:

Класс — 1

Маша пошла в магазин. Там она купила тетрадку.

Класс — 0

Маша пошла в магазин. Пингвин загорал на солнце.

Byte-pair encoding (BPE)

Byte-pair encoding — способ работы с большим словарём.

Алгоритм BPE (обучение):

1. Исходный словарь — множество символов корпуса, исходный набор правил — пустое множество
2. На каждой итерации добавляем в словарь самую часто совстречаемую в корпусе пару двух элементов словаря a, b и правило $\{a\ b \to ab\}$

Алгоритм BPE (применение): последовательно применяем каждое из полученных правил.

---

[1] Gage (C Users Journal 1994), A New Algorithm for Data Compression

[2] Sennrich et al (ACL 2016), Neural Machine Translation of Rare Words with Subword Units

BPE на примере

Проведём 5 итераций обучения алгоритма на предложении «she sells seashells by the seashore»

1. s h e | s e l l s | s e a s h e l l s | b y | t h e | s e a s h o r e

BPE на примере

Проведём 5 итераций обучения алгоритма на предложении «she sells seashells by the seashore»

1. s h e | s e l l s | s e a s h e l l s | b y | t h e | s e a s h o r e
   $\{\text{s h} \rightarrow \text{sh}\}$
2. sh e | s e l l s | s e a sh e l l s | b y | t h e | s e a sh o r e

BPE на примере

Проведём 5 итераций обучения алгоритма на предложении «she sells seashells by the seashore»

1. s h e | s e l l s | s e a s h e l l s | b y | t h e | s e a s h o r e
   $\{\text{s h} \rightarrow \text{sh}\}$
2. sh e | s e l l s | s e a sh e l l s | b y | t h e | s e a sh o r e
   $\{\text{s h} \rightarrow \text{sh}, \text{s e} \rightarrow \text{se}\}$
3. sh e | se l l s | se a sh e l l s | b y | t h e | se a sh o r e

BPE на примере

Проведём 5 итераций обучения алгоритма на предложении «she sells seashells by the seashore»

1. s h e | s e l l s | s e a s h e l l s | b y | t h e | s e a s h o r e
   {s h → sh}
2. sh e | s e l l s | s e a sh e l l s | b y | t h e | s e a sh o r e
   {s h → sh, s e → se}
3. sh e | se l l s | se a sh e l l s | b y | t h e | se a sh o r e
   {s h → sh, s e → se, l l → ll}
4. sh e | se ll s | se a sh e ll s | b y | t h e | se a sh o r e

BPE на примере

Проведём 5 итераций обучения алгоритма на предложении «she sells seashells by the seashore»

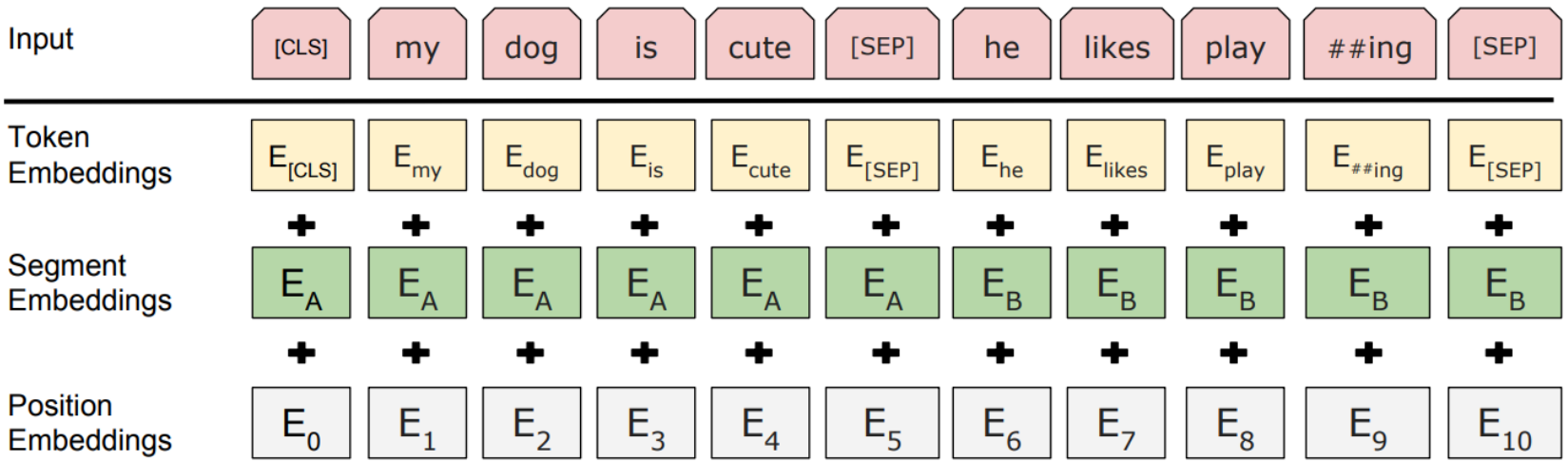
1. s h e | s e l l s | s e a s h e l l s | b y | t h e | s e a s h o r e
   {s h → sh}
2. sh e | s e l l s | s e a sh e l l s | b y | t h e | s e a sh o r e
   {s h → sh, s e → se}
3. sh e | se l l s | se a sh e l l s | b y | t h e | se a sh o r e
   {s h → sh, s e → se, l l → ll}
4. sh e | se ll s | se a sh e ll s | b y | t h e | se a sh o r e
   {s h → sh, s e → se, l l → ll, se a → sea}
5. sh e | se ll s | sea sh e ll s | b y | t h e | sea sh o r e

BERT[1] (Bidirectional Encoder Representations from Transformers)

[1] Devlin et al (2018) BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

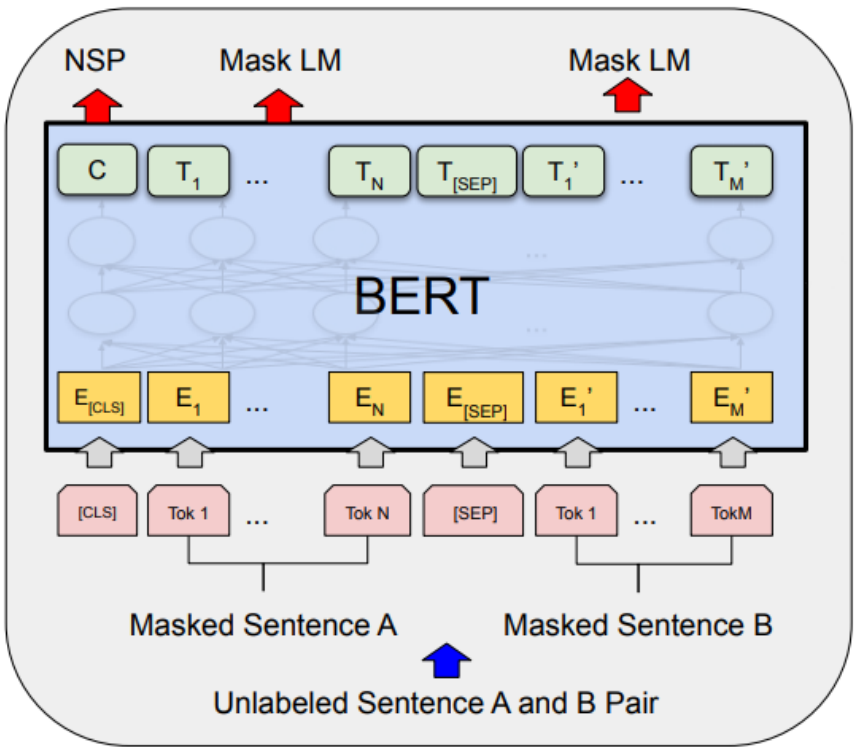
BERT: что на входе (обучение)

1. Конкатенация предложений s_i и s_j через [SEP] токен
2. Оба предложения кодируются через BPE
3. Часть токенов в предложениях маскируется
4. В начало последовательности добавляется [CLS] токен

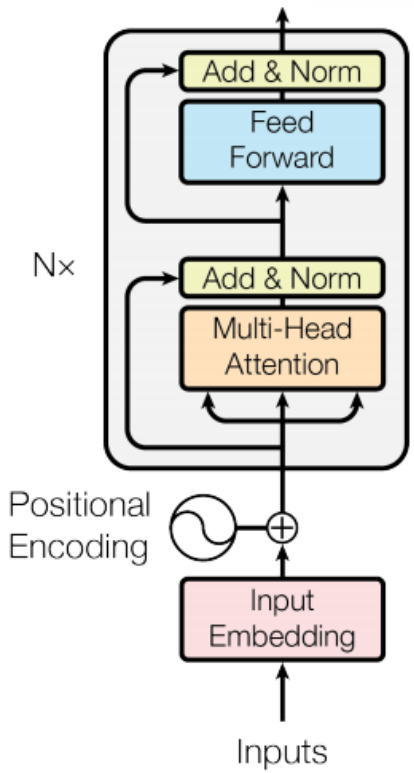


BERT: что на выходе (обучение)

1. Предсказываем $\mathbb{I}[j = i + 1]$ на выходе от [CLS] токена
2. Предсказываем правильное слово на выходе от маскированных токенов



Архитектура BERT



- Основа архитектуры — кодировщик трансформера[1]
- Две конфигурации: base и large
- base: L=12, H=768, A=12, общее число параметров ≈ 110M
- large: L=24, H=1024, A=16, общее число параметров ≈ 340M
- На выходе дополнительный линейный слой

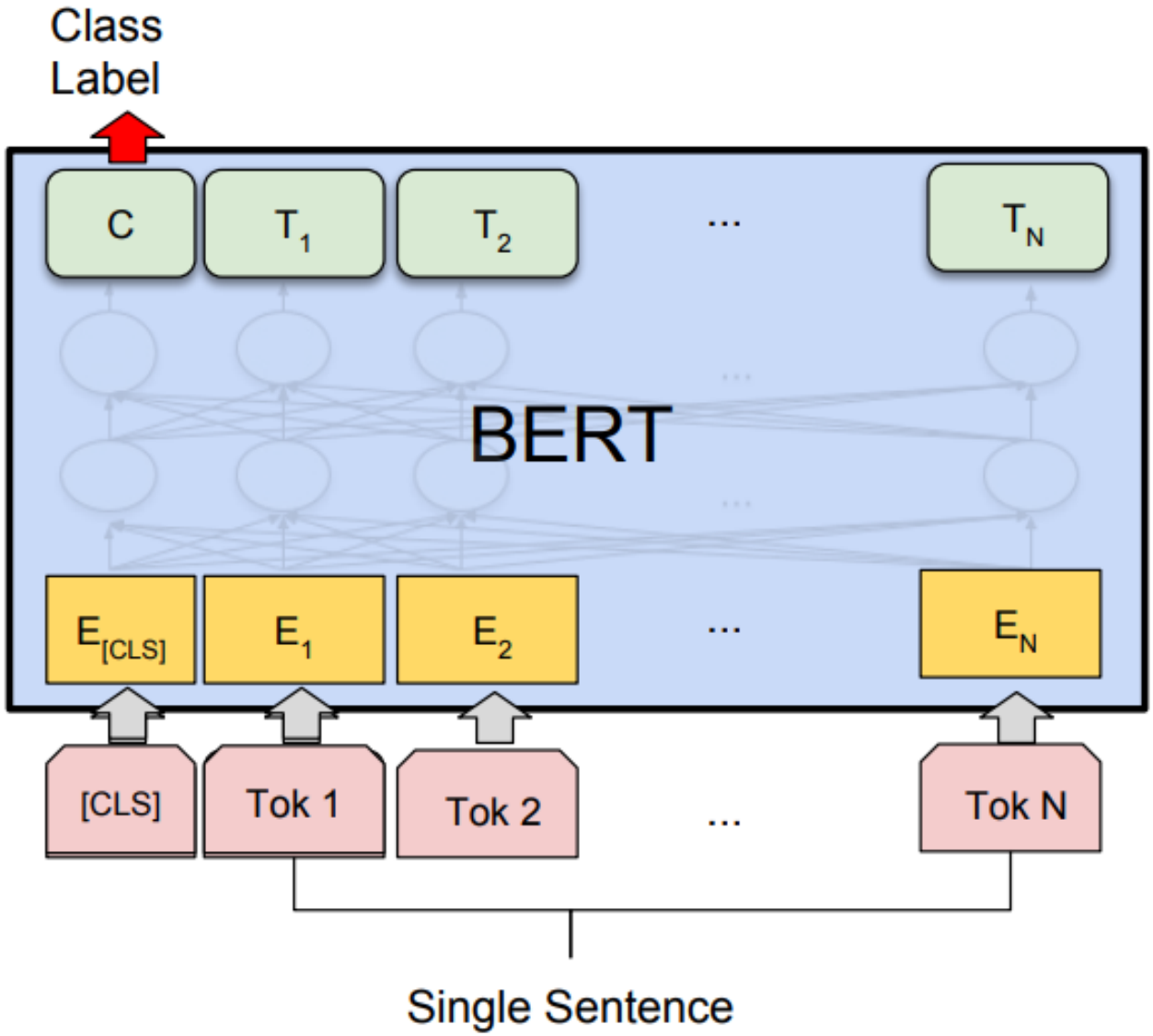
[1] Vaswani et al (NIPS 2017), Attention Is All You Need

BERT: применение

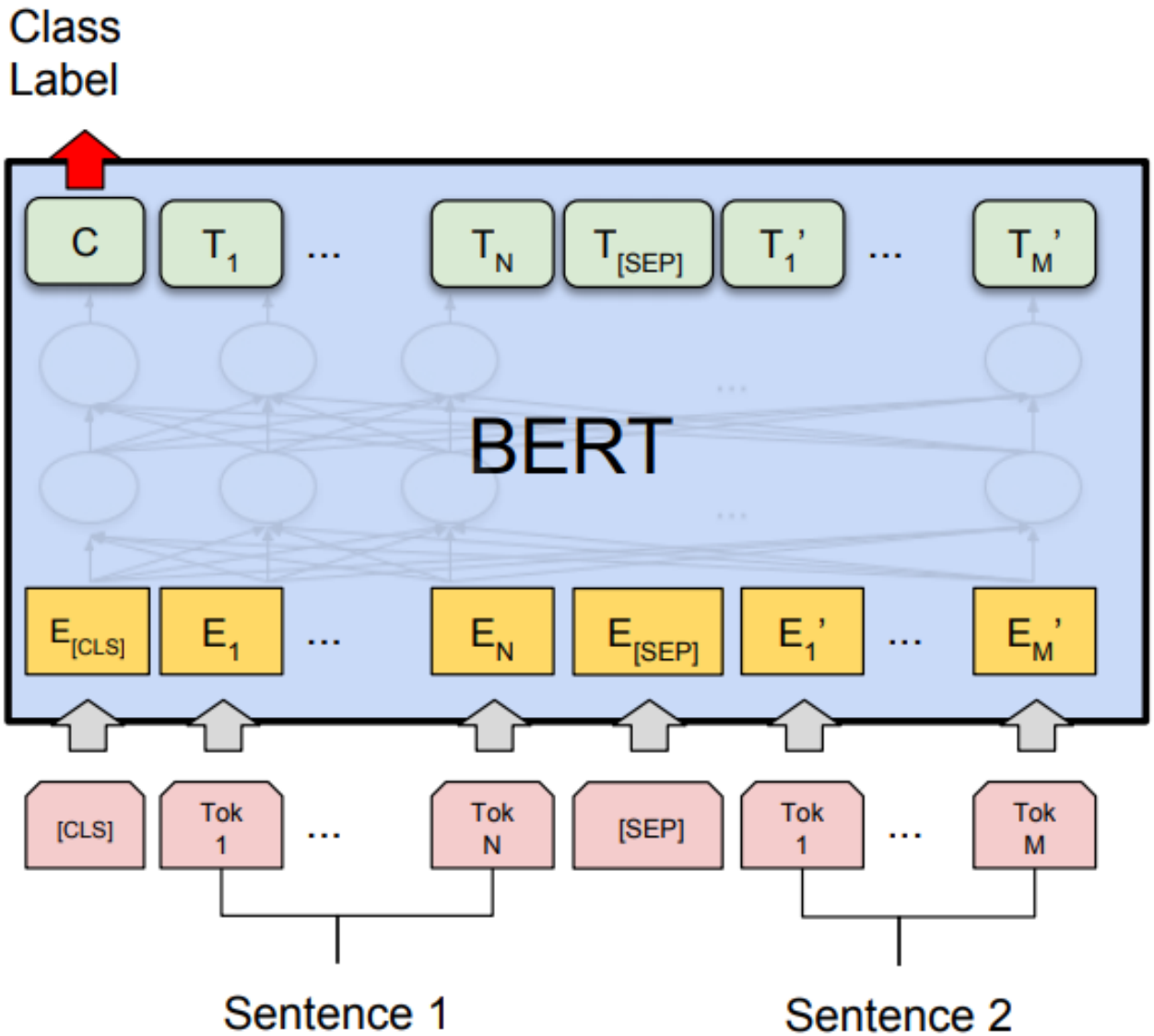
Использование BERT для transfer learning — дообучение (fine-tuning) всей сети:

1. В зависимости от приложения на вход подаётся последовательность специального вида
2. В зависимости от приложения на выходе последовательность специального вида
3. Дообучаем сеть под задачу

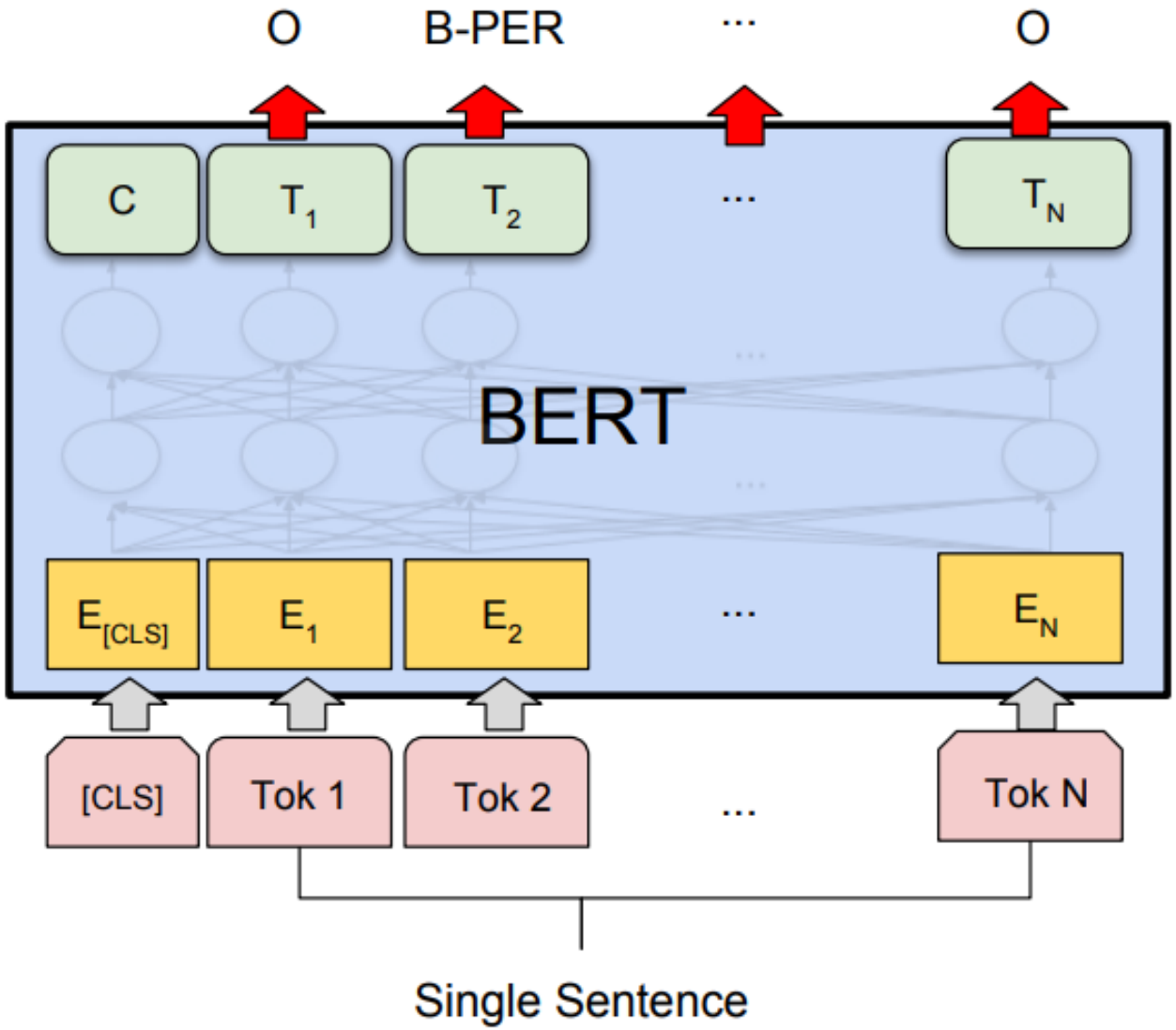
Применение BERT: классификация предложений



Применение BERT: детектирование парафразов



Применение BERT: разметка последовательности



Результаты BERT на задаче NER

Fine-tuning работает гораздо лучше чем feature-based:

System	Dev F1	Test F1
ELMo (Peters et al., 2018a)	95.7	92.2
CVT (Clark et al., 2018)	-	92.6
CSE (Akbik et al., 2018)	-	**93.1**
Fine-tuning approach		
$\text{BERT}_{\text{LARGE}}$	96.6	92.8
$\text{BERT}_{\text{BASE}}$	96.4	92.4
Feature-based approach ($\text{BERT}_{\text{BASE}}$)		
Embeddings	91.0	-
Second-to-Last Hidden	95.6	-
Last Hidden	94.9	-
Weighted Sum Last Four Hidden	95.9	-
Concat Last Four Hidden	96.1	-
Weighted Sum All 12 Layers	95.5	-

Результаты BERT на датасете SWAG

Задача: выбрать из четырёх вариантов правильный ответ на вопрос.

System	Dev	Test
ESIM+GloVe	51.9	52.7
ESIM+ELMo	59.1	59.2
OpenAI GPT	-	78.0
$\text{BERT}_{\text{BASE}}$	81.6	-
$\text{BERT}_{\text{LARGE}}$	**86.6**	**86.3**
Human (expert)†	-	85.0
Human (5 annotations)†	-	88.0

Сильное превосходство в качестве!

Результаты BERT на датасетах GLUE[1]

Сильное превосходство по всем параметрам:

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	**Average** -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

[1] https://gluebenchmark.com/leaderboard

Результат BERT на GLUE (19.10.19)

SOTA стремительно уходит вперёд:

Rank	Name	Model	URL	Score	CoLA	SST-2	MRPC	STS-B	QQP	MNLI-m	MNLI-mm	QNLI	RTE
1	ALBERT-Team Google Language	ALBERT (Ensemble)		89.4	69.1	97.1	93.4/91.2	92.5/92.0	74.2/90.5	91.3	91.0	99.2	89.2
2	王玮	ALICE v2 large ensemble (Alibaba DAMO NLP)		89.0	69.2	97.1	93.6/91.5	92.7/92.3	74.4/90.7	90.7	90.2	99.2	87.3
3	Microsoft D365 AI & UMD	FreeLB-RoBERTa (ensemble)		88.8	68.0	96.8	93.1/90.8	92.4/92.2	74.8/90.3	91.1	90.7	98.8	88.7
4	Facebook AI	RoBERTa		88.5	67.8	96.7	92.3/89.8	92.2/91.9	74.3/90.2	90.8	90.2	98.9	88.2
5	XLNet Team	XLNet-Large (ensemble)		88.4	67.8	96.8	93.0/90.7	91.6/91.1	74.2/90.3	90.2	89.8	98.6	86.3
6	Microsoft D365 AI & MSR AI	MT-DNN-ensemble		87.6	68.4	96.5	92.7/90.3	91.1/90.7	73.7/89.9	87.9	87.4	96.0	86.3
7	GLUE Human Baselines	GLUE Human Baselines		87.1	66.4	97.8	86.3/80.8	92.7/92.6	59.5/80.4	92.0	92.8	91.2	93.6
8	Stanford Hazy Research	Snorkel MeTaL		83.2	63.8	96.2	91.5/88.5	90.1/89.7	73.1/89.9	87.6	87.2	93.9	80.9
9	XLM Systems	XLM (English only)		83.1	62.9	95.6	90.7/87.1	88.8/88.2	73.2/89.8	89.1	88.5	94.0	76.0
10	Zhuosheng Zhang	SemBERT		82.9	62.3	94.6	91.2/88.3	87.8/86.7	72.8/89.8	87.6	86.3	94.6	84.5
11	Danqi Chen	SpanBERT (single-task training)		82.8	64.3	94.8	90.9/87.9	89.9/89.1	71.9/89.5	88.1	87.7	94.3	79.0
12	Kevin Clark	BERT + BAM		82.3	61.5	95.2	91.3/88.3	88.6/87.9	72.5/89.7	86.6	85.8	93.1	80.4
13	Nitish Shirish Keskar	Span-Extractive BERT on STILTs		82.3	63.2	94.5	90.6/87.6	89.4/89.2	72.2/89.4	86.5	85.8	92.5	79.8
14	Jason Phang	BERT on STILTs		82.0	62.1	94.3	90.2/86.6	88.7/88.3	71.9/89.4	86.4	85.6	92.7	80.1
15	廖亿	RGLM-Base (Huawei Noah's Ark Lab)		81.3	56.9	94.2	90.7/87.7	89.7/89.1	72.2/89.4	86.1	85.4	92.1	78.5
16	Jacob Devlin	BERT: 24-layers, 16-heads, 1024-hidden		80.5	60.5	94.9	89.3/85.4	87.6/86.5	72.1/89.3	86.7	85.9	92.7	70.1
17	Neil Houlsby	BERT + Single-task Adapters		80.2	59.2	94.3	88.7/84.3	87.3/86.1	71.5/89.4	85.4	85.0	92.4	71.6

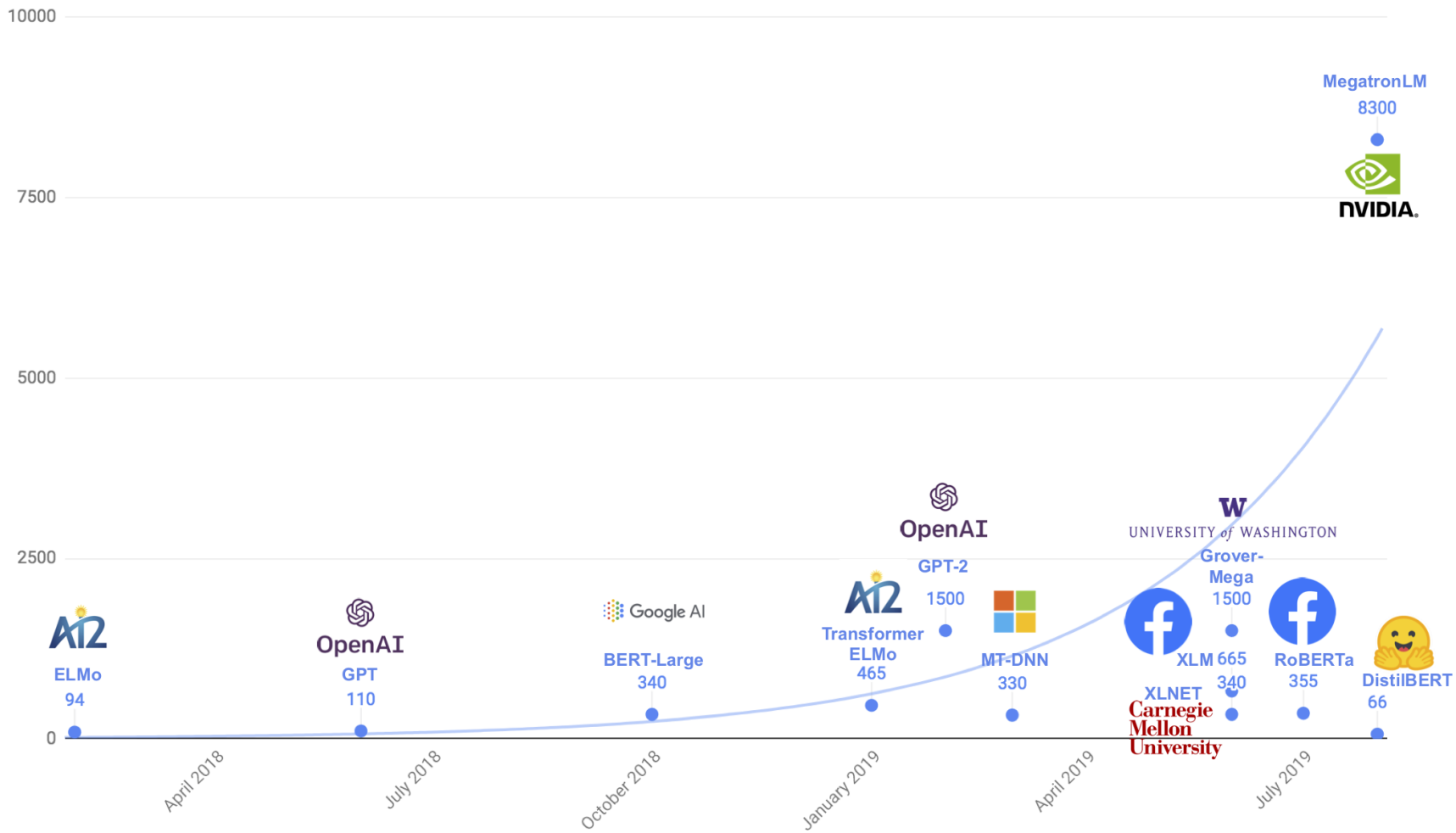
Время обучения различных моделей

ULMfit	GPT	BERT	GPT-2
Jan 2018	June 2018	Oct 2018	Feb 2019
Training:	Training	Training	Training
1 GPU day	240 GPU days	256 TPU days ~320–560 GPU days	~2048 TPU v3 days according to a reddit thread

fast.ai OpenAI Google AI OpenAI

[1] лекция CS224N/Ling284

Количество параметров в моделях



За счёт чего модели улучшаются?

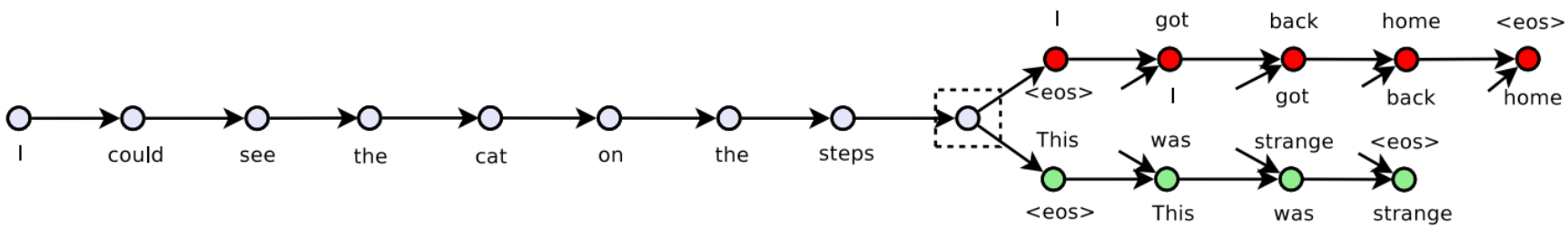
- Более долгое и правильное обучение (RoBERTa)
- Увеличение числа параметров (MegatronLM, GPT-2)
- Основа архитектуры — декодер трансформера, а не энкодер (GPT-2), выигрыш в задачах языкового моделирования
- Усложнение задачи при обучении: перемешивание слов в предложении (XLNET)
- Использование multitask learning (ERNIE 2.0)
- Упрощение архитектуры: использование одного набора параметров для слоёв различной глубины (ALBERT)

Наблюдения по BERT

- Используйте BERT в любых задачах, где вам на вход поступает предложение
- BERT гораздо лучше работает в режиме fine-tuning чем в режиме fixed-representations.
- Реализация BERT на pytorch: библиотека transformers
- Есть мультиязычный BERT (google) и мультиязычный BERT, затюненный под русский язык (deeppavlov)

Модель Skip-thoughts[1]

- В последовательном тексте по i-ому предложению необходимо восстановить $i-1$ и $i+1$ предложения
- Модель типа энкодер-декодер
- Можно использовать последнее скрытое состояние в качестве эмбеддинга



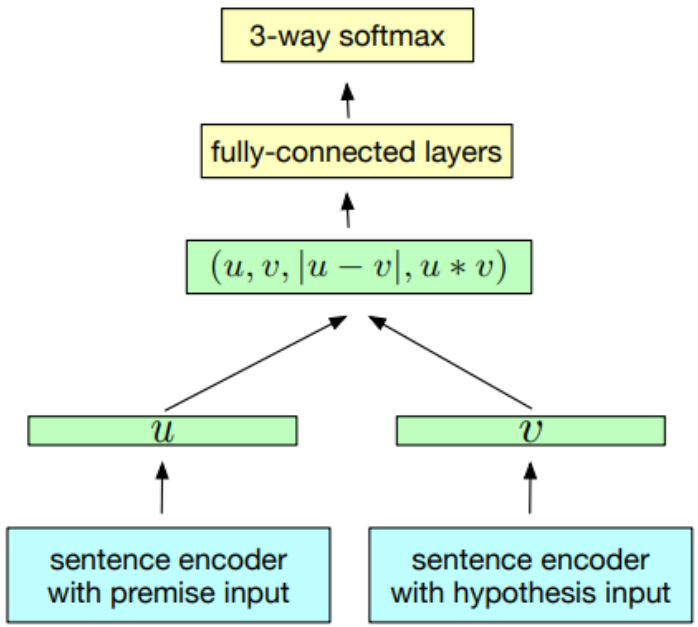
Долгое обучение, не всегда хорошее качество...[2]

[1] Kiros et al (NIPS 2015), Skip-Thought Vectors

[2] Wieting et al (ICLR 2019), No Training Required: Exploring Random Encoders for Sentence Classification

Модель Infersent

- Учится supervised на задаче SNLI (классификация пар предложений)
- Учится как сиамская сеть, энкодер для предложения — biLSTM



[1] Conneau et al (2018), Supervised Learning of Universal Sentence Representations from Natural Language Inference Data

Universal Sentence Encoder

- Учится unsupervised как skip-thoughts, но с использованием трансформера
- Дообучен под задачу определения парафразов (датасет STS)

Approach	CR	MPQA	MR	MRPC	SICK-E	SST-2	SST-5
Baseline							
Random Embedding	61.16	68.41	48.75	64.35	54.94	49.92	24.48
Experiments							
ELMo (BoW, all layers, 5.5B)	83.95	**91.02**	**80.91**	72.93	82.36	**86.71**	47.60
ELMo (BoW, all layers, original)	85.11	89.55	79.72	71.65	81.86	86.33	**48.73**
ELMo (BoW, top layer, original)	84.13	89.30	79.36	70.20	79.64	85.28	47.33
Word2Vec (BoW, google news)	79.23	88.24	77.44	73.28	79.09	80.83	44.25
p-mean (monolingual)	80.82	89.09	78.34	73.22	83.52	84.07	44.89
FastText (BoW, common crawl)	79.63	87.99	78.03	74.49	79.28	83.31	44.34
GloVe (BoW, common crawl)	78.67	87.90	77.63	73.10	79.01	81.55	45.16
USE (DAN)	80.50	83.53	74.03	71.77	80.39	80.34	42.17
USE (Transformer)	**86.04**	86.99	80.20	72.29	83.32	86.05	48.10
InferSent (AllNLI)	83.58	89.02	80.02	**74.55**	**86.44**	83.91	47.74
SkipThought	81.03	87.06	76.60	73.22	84.33	81.77	44.80

[1] Cer et al (2018), Universal Sentence Encoder

Резюме по лекции

- По умолчанию, в задачах, связанных с предложениями, при нехватке данных используйте BERT
- Если у вас английский язык и неспецифичный домен, используйте модификации BERT
- Модели ELMO и ULMFit могут пригодиться, если у вас специфичный язык/домен, для которого не обучено transformer-моделей
- Skip-Thoughts, InferSent, USE могут быть полезные в некоторых приложениях (например, вычисление расстояния между предложениями)