



**girafe
ai**

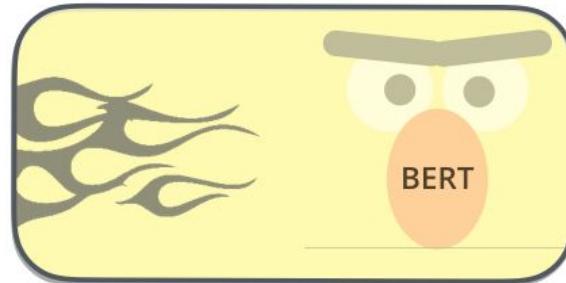
Lecture 06: ULMFiT and Transformer-XL, Question Answering

Radoslav Neychev

Fall 2021, Moscow, Russia

1. ULMFiT: dropout in the wild
2. Transformer-XL
3. Question Answering
 - SQuAD, SberQuAD
 - Open-Domain Question Answering
4. More on GPT
 - miniGPT and GPT-3 for Russian language

Based on: <http://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture13-contextual-representations.pdf>
<https://jalammar.github.io/illustrated-transformer/>
<http://jalammar.github.io/illustrated-bert/>
<https://medium.com/mlreview/understanding-building-blocks-of-ulmfit-818d3775325b>



ULMFiT: Universal Language Model Fine-tuning for Text Classification

ULMFiT: architecture

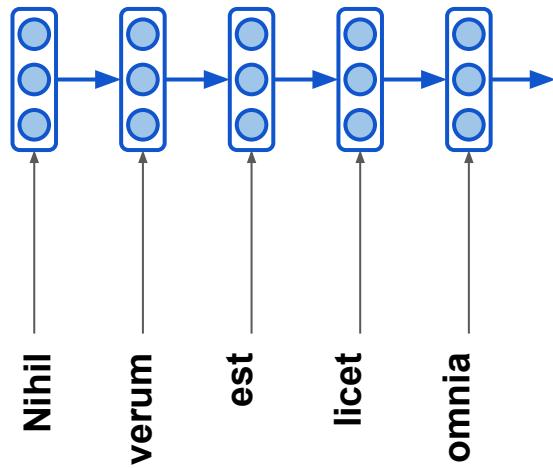
- Encoder: AWD-LSTM (ASGD Weight-Dropped LSTM)

```
SequentialRNN(  
    (0): MultiBatchEncoder(  
        (module): AWD_LSTM(  
            (encoder): Embedding(60003, 300, padding_idx=1)  
            (encoder_dp): EmbeddingDropout(  
                (emb): Embedding(60003, 300, padding_idx=1)  
            )  
            (rnns): ModuleList(  
                (0): WeightDropout(  
                    (module): LSTM(300, 1150, batch_first=True)  
                )  
                (1): WeightDropout(  
                    (module): LSTM(1150, 1150, batch_first=True)  
                )  
                (2): WeightDropout(  
                    (module): LSTM(1150, 300, batch_first=True)  
                )  
            )  
            (input_dp): RNNDropout()  
            (hidden_dps): ModuleList(  
                (0): RNNDropout()  
                (1): RNNDropout()  
                (2): RNNDropout()  
            )  
        )  
    )  
    (1): PoolingLinearClassifier(  
        (layers): Sequential(  
            (0): BatchNorm1d(900, eps=1e-05, momentum=0.1, affine=True,  
track_running_stats=True)  
            (1): Dropout(p=0.4)  
            (2): Linear(in_features=900, out_features=50, bias=True)  
            (3): ReLU(inplace)  
            (4): BatchNorm1d(50, eps=1e-05, momentum=0.1, affine=True,  
track_running_stats=True)  
            (5): Dropout(p=0.1)  
            (6): Linear(in_features=50, out_features=2, bias=True)  
        )  
    )  
)
```

What's the main difference from all the other Sesame street models?



RNNs and dropout



ULMFiT: architecture

- Encoder: AWD-LSTM (ASGD Weight-Dropped LSTM)
- AWD-LSTM literally has dropout at all the possible layers as long as it makes sense.

```
SequentialRNN(  
    (0): MultiBatchEncoder(  
        (module): AWD_LSTM(  
            (encoder): Embedding(60003, 300, padding_idx=1)  
            (encoder_dp): EmbeddingDropout(  
                (emb): Embedding(60003, 300, padding_idx=1)  
            )  
            (rnns): ModuleList(  
                (0): WeightDropout(  
                    (module): LSTM(300, 1150, batch_first=True)  
                )  
                (1): WeightDropout(  
                    (module): LSTM(1150, 1150, batch_first=True)  
                )  
                (2): WeightDropout(  
                    (module): LSTM(1150, 300, batch_first=True)  
                )  
            )  
            (input_dp): RNNDropout()  
            (hidden_dps): ModuleList(  
                (0): RNNDropout()  
                (1): RNNDropout()  
                (2): RNNDropout()  
            )  
        )  
    )  
    (1): PoolingLinearClassifier(  
        (layers): Sequential(  
            (0): BatchNorm1d(900, eps=1e-05, momentum=0.1, affine=True,  
track_running_stats=True)  
            (1): Dropout(p=0.4)  
            (2): Linear(in_features=900, out_features=50, bias=True)  
            (3): ReLU(inplace)  
            (4): BatchNorm1d(50, eps=1e-05, momentum=0.1, affine=True,  
track_running_stats=True)  
            (5): Dropout(p=0.1)  
            (6): Linear(in_features=50, out_features=2, bias=True)  
        )  
    )  
)
```

ULMFiT: encoder dropout

1. Encoder Dropout (`EmbeddingDropout()`): Zeroing embedding vectors randomly.

ENCODER DROPOUT				
	before dropout			after dropout
token	d1	d2	...	token
I	0.399	0.75379	0.62616	I
love	0.88533	0.29449	0.15856	love
cats	0.48927	0.04071	0.21427	cats
dogs	0.72918	0.86882	0.77136	dogs

ULMFiT: input dropout

2. Input Dropout (`RNNDropout()`): Zeroing embedding

INPUT DROPOUT				
batch: [I love cats, I love dogs]				
before dropout				after dropout
I	0	0	0	
love	0.88533	0.29449	0.15856	
cats	0.48927	0.04071	0.21427	
before dropout				after dropout
I	0	0	0	
love	0.88533	0.29449	0.15856	
dogs	0.72918	0.86882	0.77136	

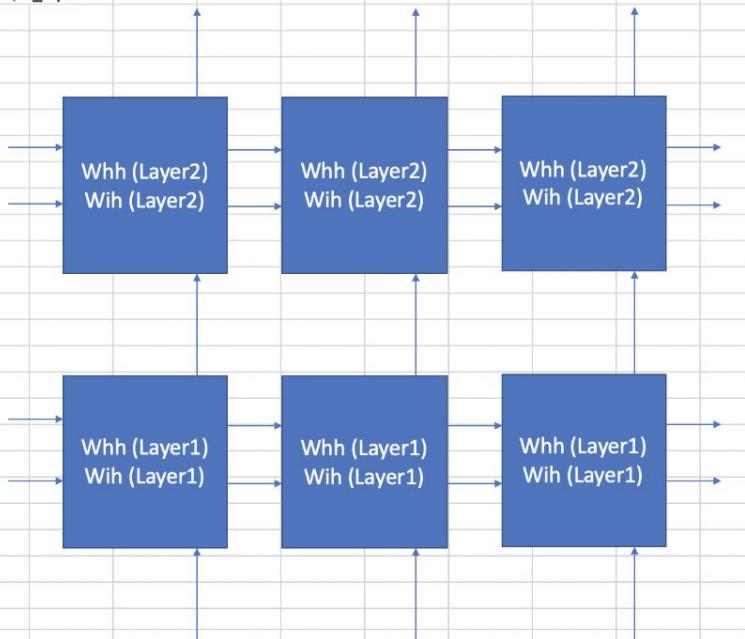
3. Weight Dropout(WeightDropout()): Apply dropout to LSTM weights.

Random dropout at hidden-to-hidden weights for each LSTM layer

```
rnn = nn.LSTM(5,10,1, bidirectional=False, batch_first=True)  
WeightDropout(rnn, weight_p=0.5, layer_names=['weight_hh_10'])
```

ULMFiT: weight dropout

WEIGHT DROPOUT
n_hid=5, n_layers=2



I	I
0	0
0	0
0	0

love	love
0	0.88533
0.29449	0.29449
0.15856	0

cats	dogs
0	0.72918
0.04071	0.86882
0.21427	0

same word, different vectors

before dropout

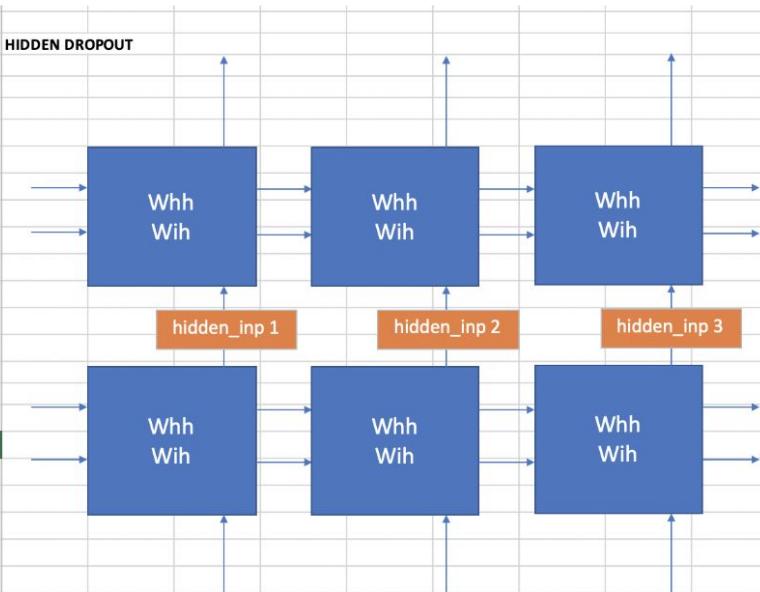
Whh (Layer1)				
0.30546	0.94294	0.66715	0.5655	0.31191
0.06246	0.36674	0.35672	0.53357	0.04652
0.84278	0.17049	0.92283	0.97827	0.67913
0.94699	0.02277	0.79537	0.09479	0.16284
0.34098	0.09256	0.69661	0.22605	0.57153
0.09455	0.78172	0.48226	0.82222	0.25151
0.92516	0.50419	0.82879	0.67049	0.84304
0.11741	0.09362	0.69367	0.5633	0.34755
0.98025	0.73131	0.21491	0.86319	0.31456
0.0289	0.90899	0.30697	0.75105	0.62107
0.96808	0.4712	0.77992	0.83567	0.09754
0.15539	0.94866	0.51977	0.51167	0.08723
0.10944	0.88881	0.74007	0.96823	0.03516
0.62014	0.4363	0.02097	0.61126	0.30466
0.10465	0.02462	0.2334	0.25372	0.54579
0.14384	0.962	0.70157	0.39285	0.41064
0.42081	0.64236	0.06941	0.41805	0.78772
0.45229	0.9871	0.49831	0.17108	0.48868
0.42142	0.66105	0.53273	0.30901	0.45095
0.33421	0.77472	0.33966	0.42693	0.4567

after dropout

Whh (Layer1)				
0.30546	0	0.66715	0.5655	0
0	0.36674	0.35672	0.53357	0.04652
0.84278	0.17049	0.92283	0.97827	0.67913
0.94699	0.02277	0.79537	0.09479	0.16284
0.34098	0.09256	0.69661	0.22605	0.57153
0	0.78172	0.48226	0.82222	0.25151
0	0.50419	0	0.67049	0.84304
0	0.09362	0.69367	0.5633	0
0.98025	0.73131	0	0	0.31456
0	0	0.30697	0.75105	0.62107
0	0	0.77992	0.83567	0.09754
0.15539	0	0.51977	0.51167	0.08723
0.10944	0.88881	0.74007	0.96823	0.03516
0.62014	0.4363	0.02097	0.61126	0.30466
0.10465	0.02462	0.2334	0	0.54579
0	0.962	0.70157	0	0.41064
0	0	0.06941	0.41805	0.78772
0.45229	0.9871	0.49831	0	0.48868
0.42142	0.66105	0.53273	0.30901	0.45095
0	0.77472	0.33966	0.42693	0.4567

ULMFiT: hidden dropout

4. Hidden Dropout (RNNDropout ()): Zeroing outputs of LSTM layers. This dropout is applied except for the last LSTM layer output.



before dropout										
I	0.11587	0.89354	0.18537	0.13159	0.04328	0.36997	0.68241	0.21505	0.07696	0.1405
love	0.51854	0.94156	0.80541	0.49411	0.44335	0.31453	0.20678	0.95815	0.70224	0.2766
cats	0.41255	0.70882	0.76667	0.93813	0.89034	0.1676	0.92944	0.32216	0.11675	0.99425

after dropout										
hidden_inp 1	I	0.11587	0	0.18537	0.13159	0	0.36997	0.68241	0	0
hidden_inp 2	love	0.51854	0	0.80541	0.49411	0	0.31453	0.20678	0	0
hidden_inp 3	cats	0.41255	0	0.76667	0.93813	0	0.1676	0.92944	0	0

before dropout										
I	0.90644	0.01673	0.38366	0.86887	0.26845	0.332	0.93804	0.34769	0.83711	0.66387
love	0.82539	0.09143	0.32444	0.35099	0.64491	0.79199	0.24797	0.56466	0.46948	0.33358
dogs	0.38805	0.56548	0.1424	0.31113	0.11645	0.36759	0.73068	0.05727	0.7315	0.01498

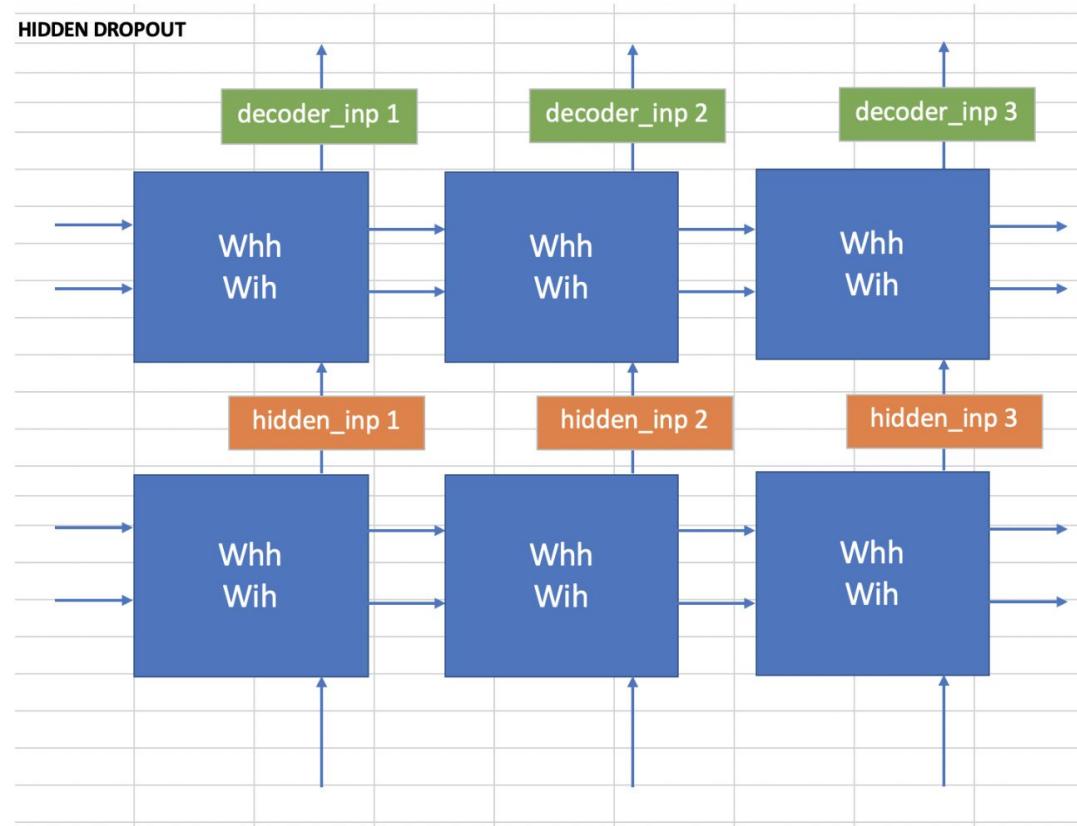
after dropout										
hidden_inp 1	I	0	0	0.38366	0.86887	0	0.332	0	0.34769	0.83711
hidden_inp 2	love	0	0	0.32444	0.35099	0	0.79199	0	0.56466	0.46948
hidden_inp 3	dogs	0	0	0.1424	0.31113	0	0.36759	0	0.05727	0.7315

ULMFiT: output dropout

5. Output Dropout

(`RNNDropout()`):

Zeroing final sequence outputs from encoder before feeding it to decoder.



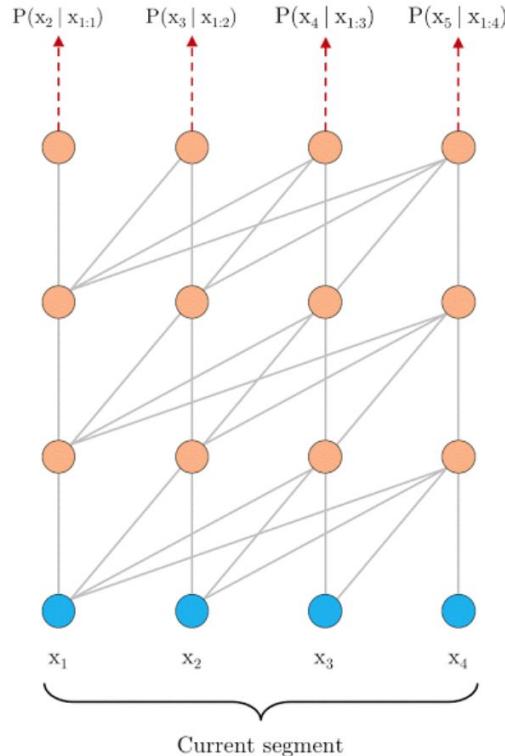
Transformer-XL

- Vanilla Transformer works with a fixed-length context at training time. That's why:
 - the algorithm is not able to model dependencies that are longer than a fixed length.
 - the segments usually do not respect the sentence boundaries, resulting in context fragmentation which leads to inefficient optimization.

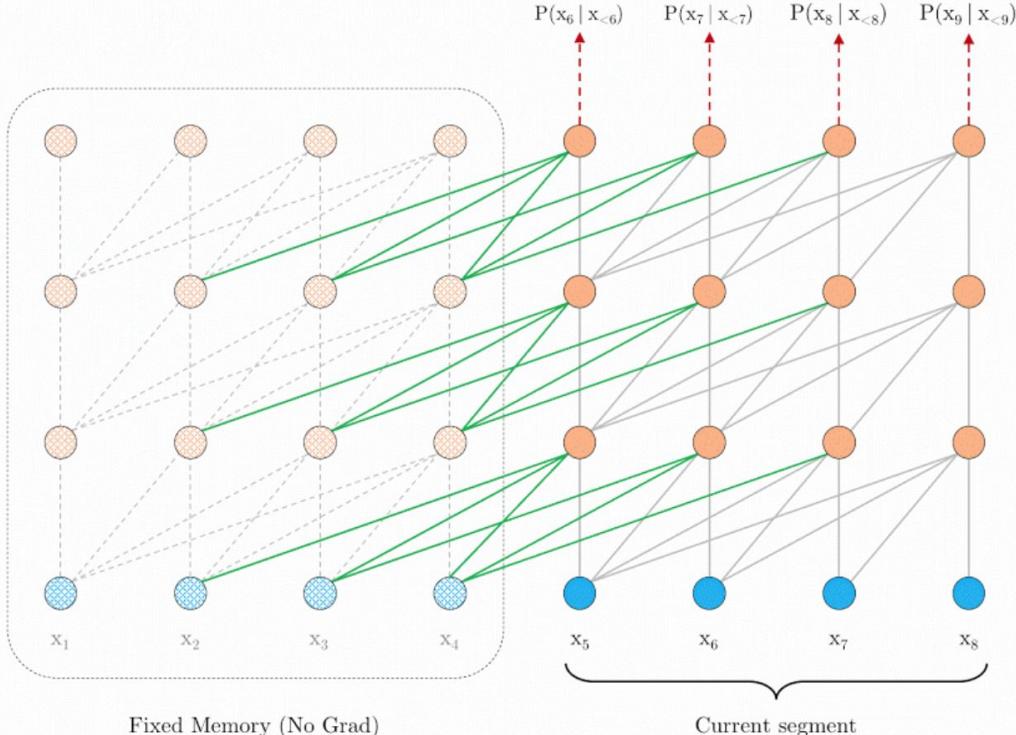
Segment-level Recurrence

- During training, the representations computed for the previous segment are fixed and cached to be reused as an extended context when the model processes the next new segment.
- Contextual information is now able to flow across segment boundaries.
- Recurrence mechanism also resolves the context fragmentation issue, providing necessary context for tokens in the front of a new segment.

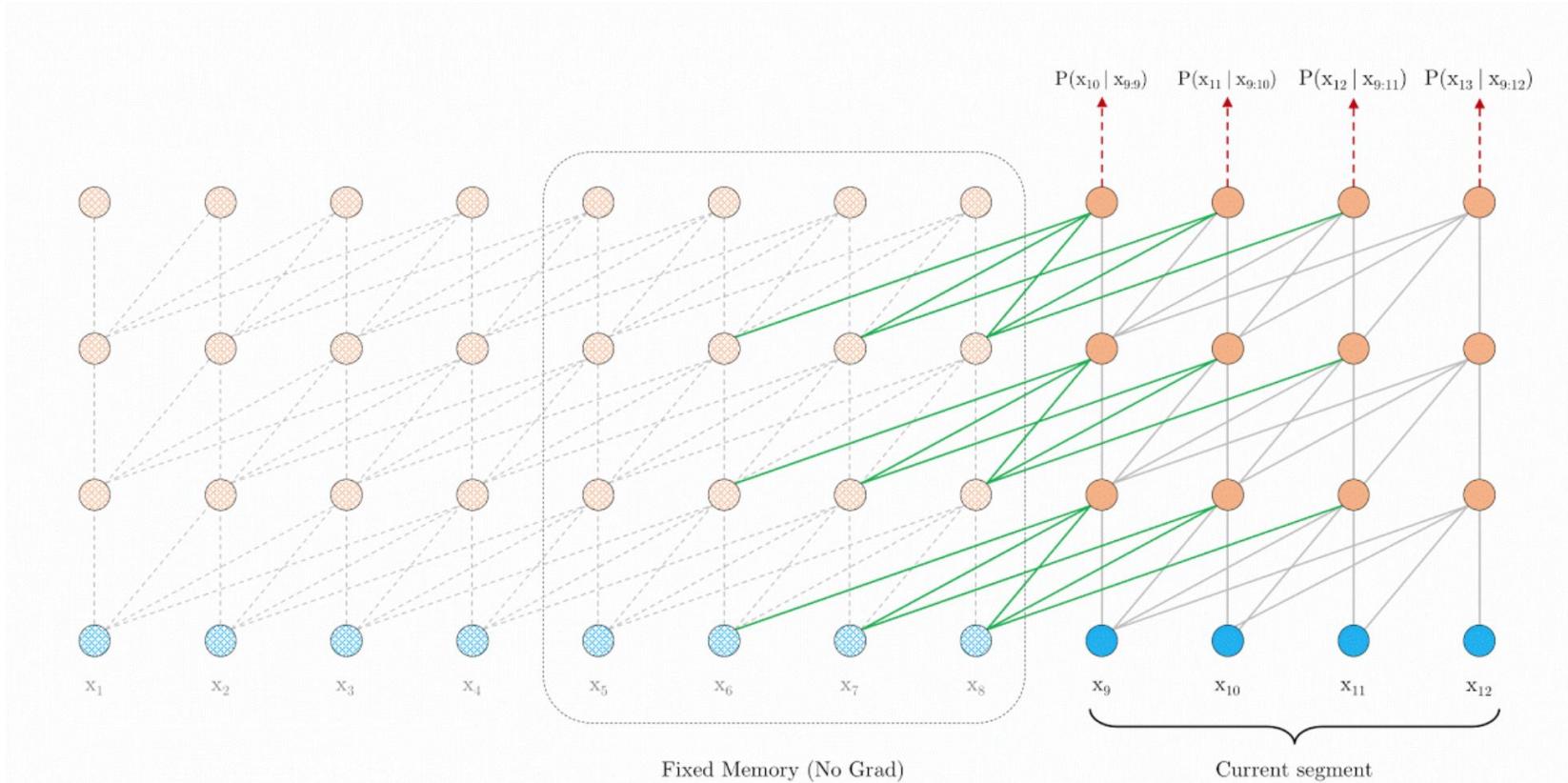
Segment-level Recurrence



Segment-level Recurrence



Segment-level Recurrence



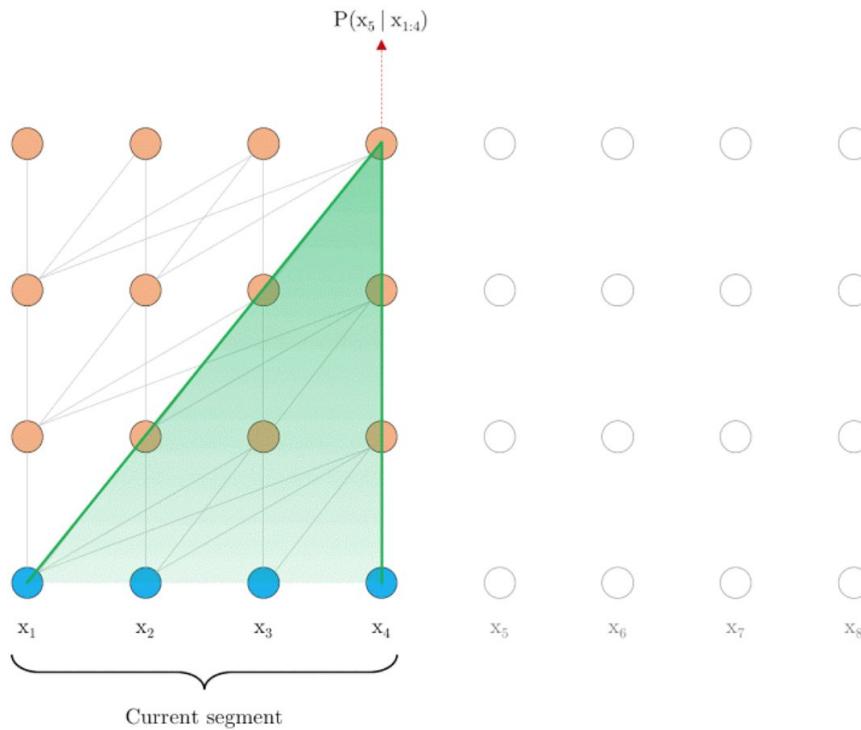
Relative Positional Encodings

- Fixed embeddings with learnable transformations instead of learnable embeddings

As a result:

- more generalizable to longer sequences at test time
- longer effective context

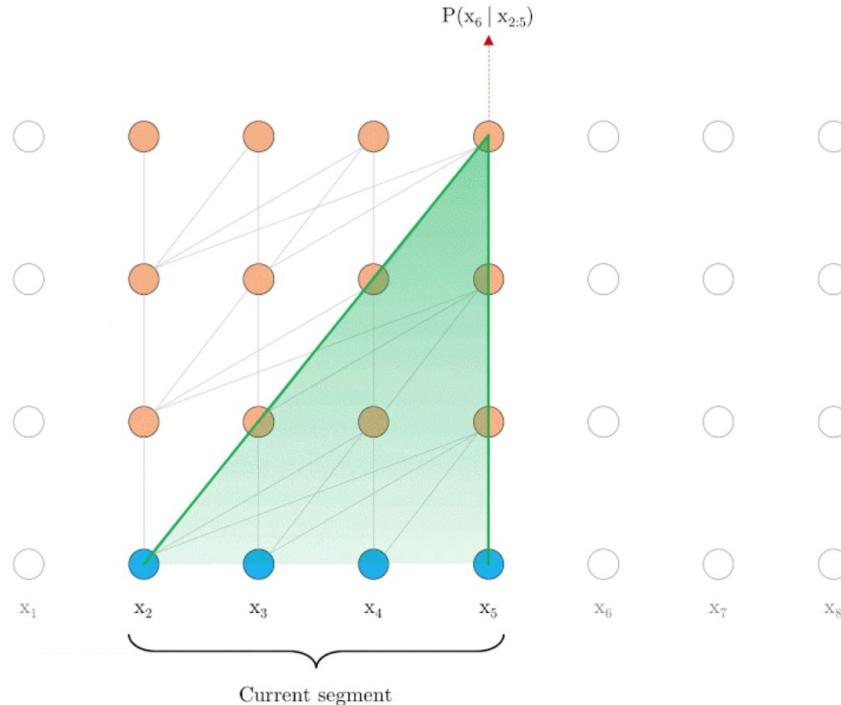
Vanilla Transformer vs. Transformer-XL



Vanilla Transformer with a fixed-length context at evaluation time

source: [Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context](#)

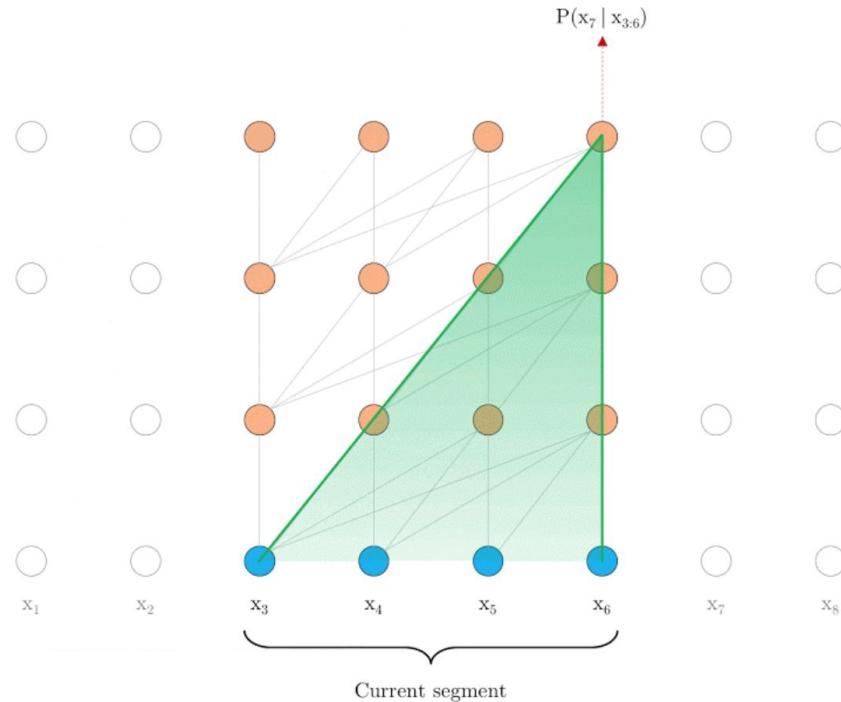
Vanilla Transformer vs. Transformer-XL



Vanilla Transformer with a fixed-length context at evaluation time

source: [Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context](#)

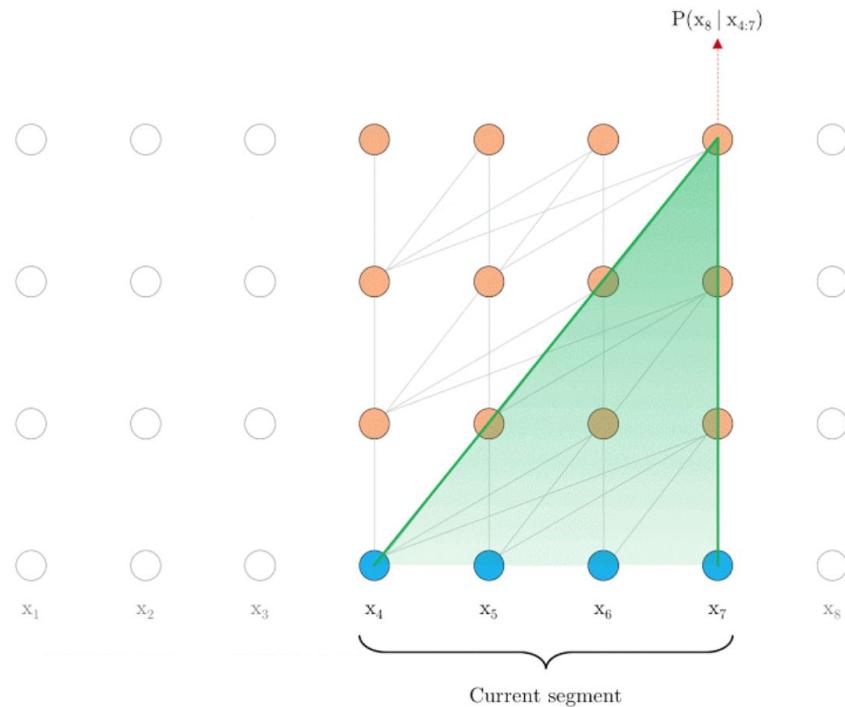
Vanilla Transformer vs. Transformer-XL



Vanilla Transformer with a fixed-length context at evaluation time

source: [Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context](#)

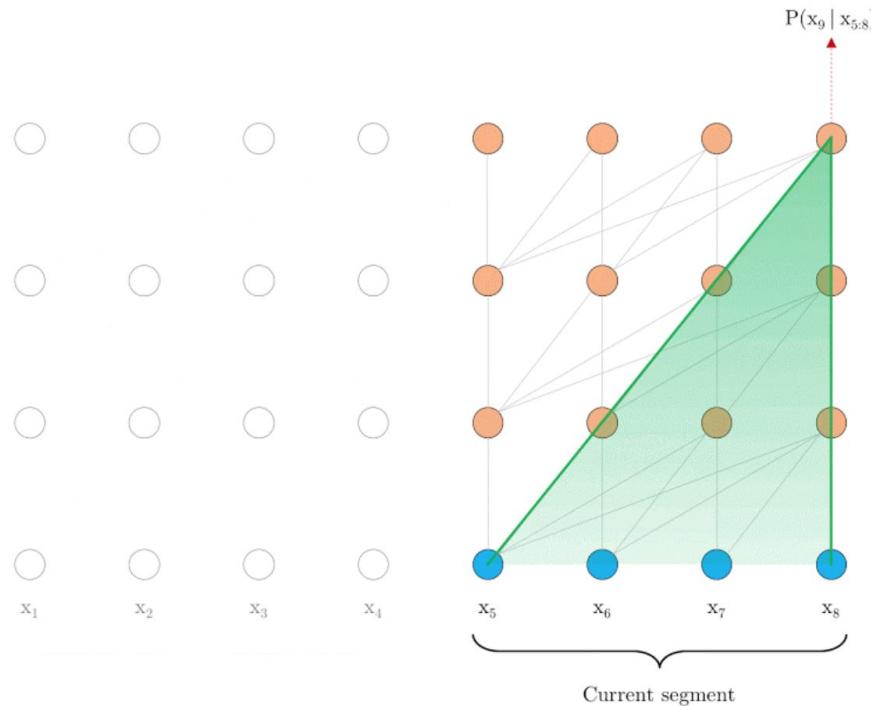
Vanilla Transformer vs. Transformer-XL



Vanilla Transformer with a fixed-length context at evaluation time

source: [Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context](#)

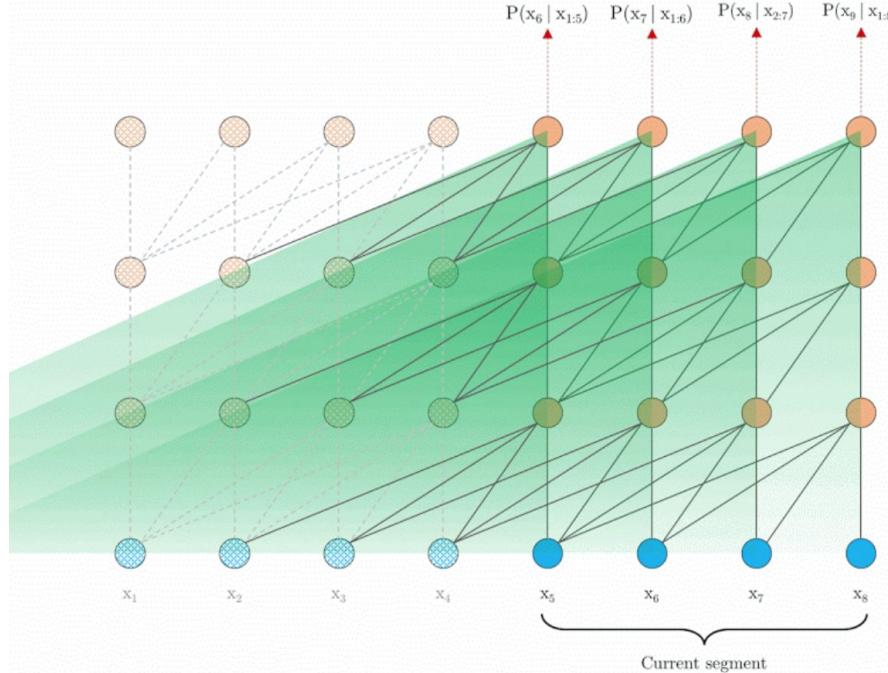
Vanilla Transformer vs. Transformer-XL



Vanilla Transformer with a fixed-length context at evaluation time

source: [Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context](#)

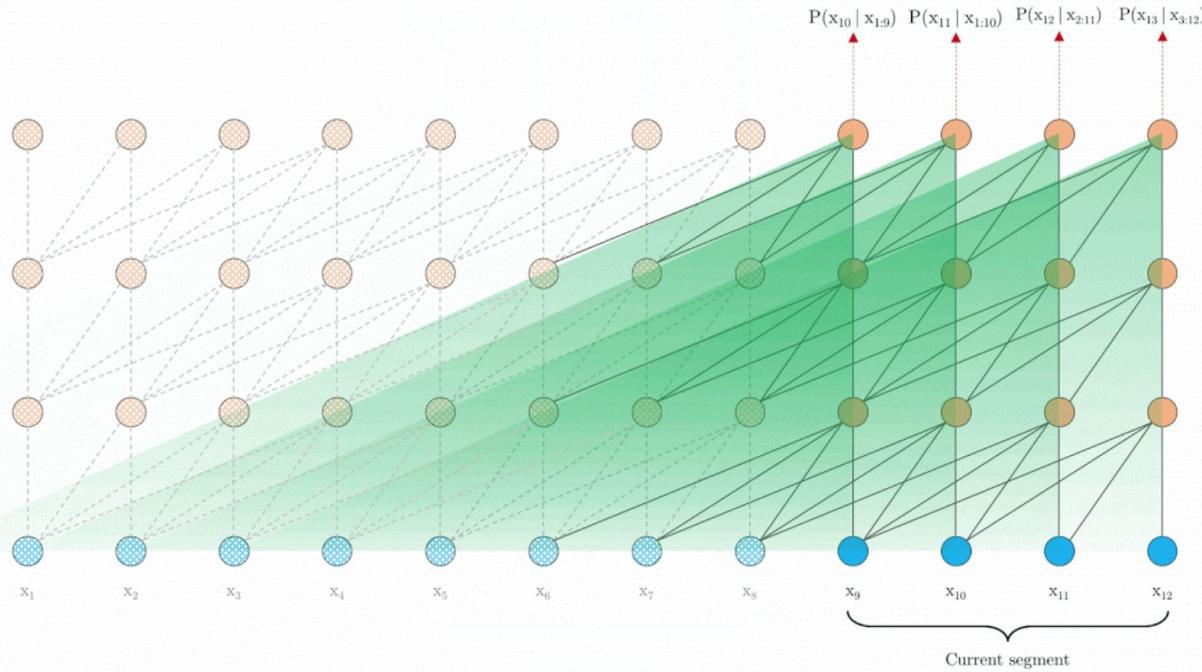
Vanila Transformer vs. Transformer-XL



Transformer-XL with segment-level recurrence at evaluation time

source: [Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context](#)

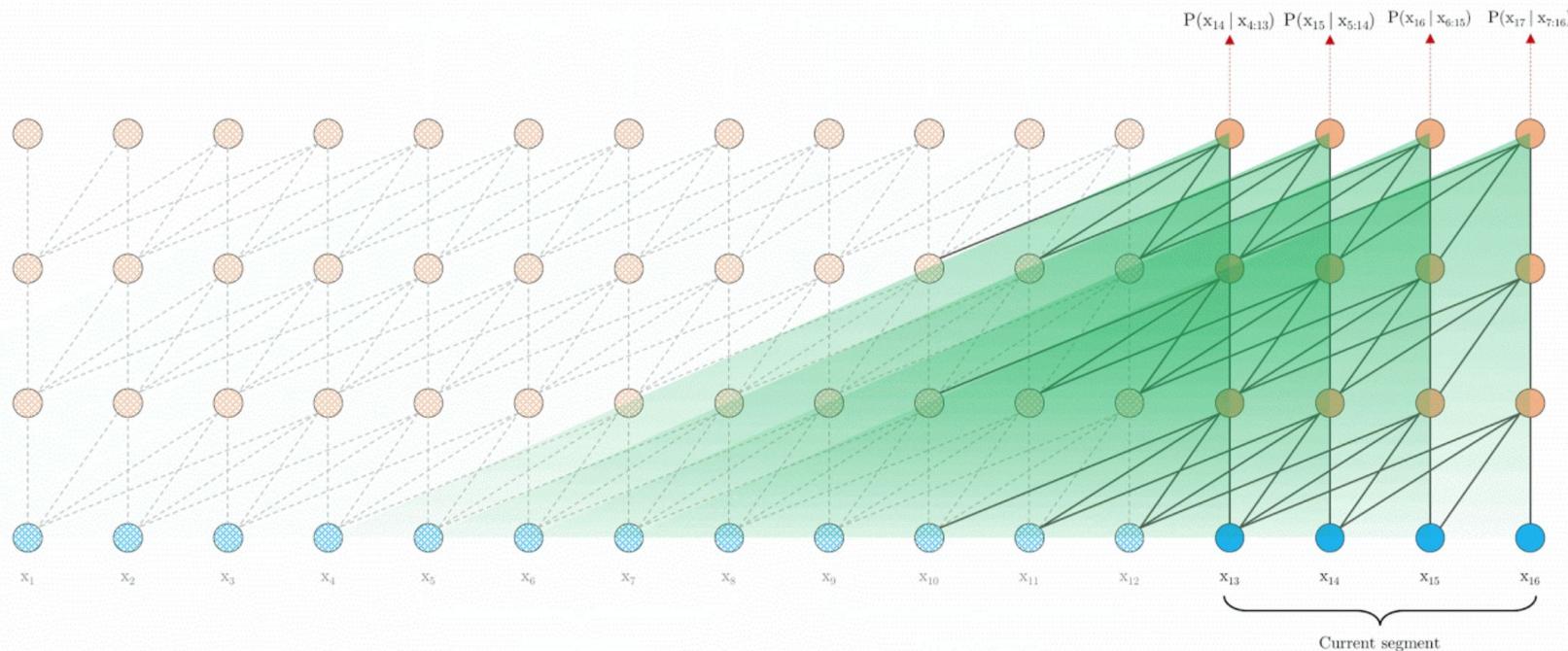
Vanila Transformer vs. Transformer-XL



Transformer-XL with segment-level recurrence at evaluation time

source: [Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context](#)

Vanila Transformer vs. Transformer-XL



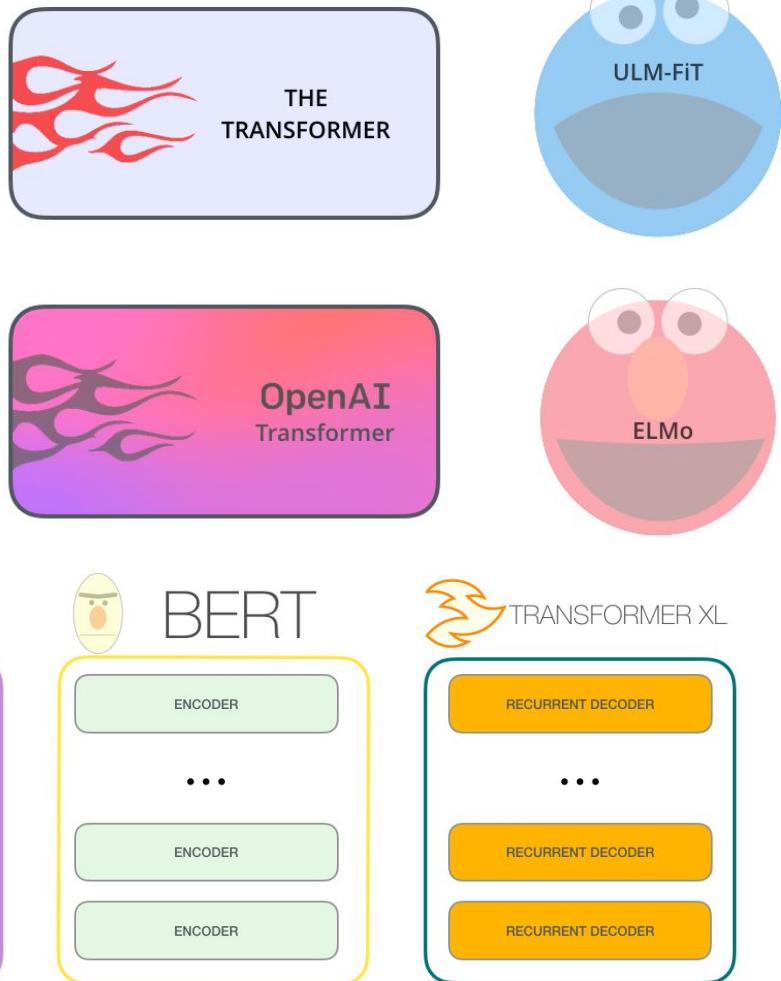
Transformer-XL with segment-level recurrence at evaluation time

source: [Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context](#)

Vanila Transformer vs. Transformer-XL

- Transformer-XL learns dependency that is about 80% longer than RNNs and 450% longer than vanilla Transformers
- Transformer-XL is up to 1,800+ times faster than a vanilla Transformer during evaluation on language modeling tasks, because no re-computation is needed

- Transformer
- OpenAI Transformer
- ELMO
- ULMFiT
- Transformer-XL
- BERT
- BERTology
- GPT
- GPT-2
- GPT-3



Question Answering

A Brief History of Open-domain Question Answering

- Simmons et al. (1964) did first exploration of answering questions from an expository text based on matching dependency parses of a question and answer
- Murax (Kupiec 1993) aimed to answer questions over an online encyclopedia using IR and shallow linguistic processing
- The NIST TREC QA track begun in 1999 first rigorously investigated answering fact questions over a large collection of documents
- IBM's Jeopardy! System (DeepQA, 2011) brought attention to a version of the problem; it used an ensemble of many methods
- DrQA (Chen et al. 2016) uses IR followed by neural reading comprehension to bring deep learning to Open-domain QA

MCTest Reading Comprehension

Passage (P)

+

Question (Q)



Answer (A)

P
Alyssa got to the beach after a long trip. She's from Charlotte. She traveled from Atlanta. She's now in Miami. She went to Miami to visit some friends. But she wanted some time to herself at the beach, so she went there first. After going swimming and laying out, she went to her friend Ellen's house. Ellen greeted Alyssa and they both had some lemonade to drink. Alyssa called her friends Kristin and Rachel to meet at Ellen's house.....

Q Why did Alyssa go to Miami?

A To visit some friends

Stanford Question Answering Dataset (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**

SQuAD evaluation, v1.1

- Authors collected 3 gold answers
- Systems are scored on two metrics:
 - Exact match: 1/0 accuracy on whether you match one of the 3 answers
 - F1: Take system and each gold answer as bag of words, evaluate Precision, Recall and harmonic mean F1.
Score is (macro-)average of per-question F1 scores
- F1 measure is seen as more reliable and taken as primary
 - It's less based on choosing exactly the same span that humans chose, which is susceptible to various effects, including line breaks
- Both metrics ignore punctuation and articles (**a, an, the** only)

SQuAD v1.1 leaderboard, end of 2016

		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
--	------	------

SQuAD v1.1 leaderboard, (May 2020)

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

- A defect of SQuAD 1.0 is that all questions have an answer in the paragraph
- Systems (implicitly) rank candidates and choose the best one
- You don't have to judge whether a span answers the question
- In SQuAD 2.0, 1/3 of the training questions have no answer, and about 1/2 of the dev/test questions have no answer
 - For NoAnswer examples, NoAnswer receives a score of 1, and any other response gets 0, for both exact match and F1
- Simplest system approach to SQuAD 2.0:
 - Have a threshold score for whether a span answers a question
- Or you could have a second component that confirms answering
 - Like Natural Language Inference (NLI) or “Answer validation”

SQuAD 2.0 example

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 nInet]

SQuAD 2.0 leaderboard (October 2020)

Rank	Model	EM	F1
	Human Performance <i>Stanford University</i> <i>(Rajpurkar & Jia et al. '18)</i>	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) <i>Shanghai Jiao Tong University</i> 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-Sinotic	90.386	92.777
5 Jan 10, 2020	Retro-Reader on ALBERT (ensemble) <i>Shanghai Jiao Tong University</i> http://arxiv.org/abs/2001.09694	90.115	92.580
5 Sep 27, 2020	electra+nlayers (ensemble) oppo.tensorlab	90.126	92.535

Now in Russian: SberQuAD

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

SberQuAD evaluation

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.

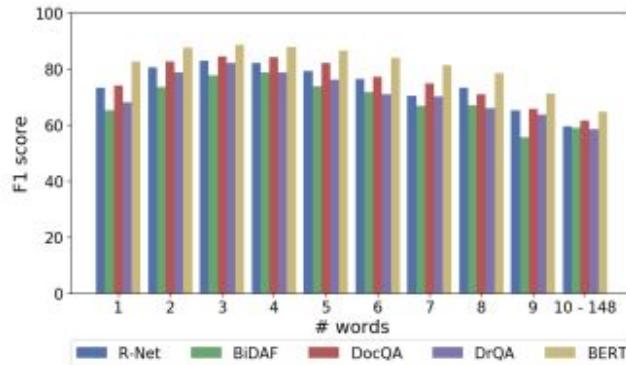


Figure 6: Model performance depending on answer length (# of words).

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

Table 8: Answer quality for misspelled questions.

But errors are still present

The Yuan dynasty is considered both a successor to the Mongol Empire and an imperial Chinese dynasty. It was the khanate ruled by the successors of Möngke Khan after the division of the Mongol Empire. In official Chinese histories, the Yuan dynasty bore the Mandate of Heaven, following the Song dynasty and preceding the Ming dynasty. The dynasty was established by Kublai Khan, yet he placed his grandfather Genghis Khan on the imperial records as the official founder of the

What dynasty came before the Yuan?

Gold Answers: ① Song dynasty ② Mongol Empire
③ the Song dynasty

Prediction: Ming dynasty [BERT (single model) (Google AI)]

S(ber)QuAD limitations

- Only span-based answers (no yes/no, counting, implicit why)
 - Questions were constructed looking at the passages
 - Not genuine information needs
 - Generally greater lexical and syntactic matching between questions and answer span than you get IRL
 - Barely any multi-fact/sentence inference beyond coreference
- But these datasets are still of a great use

Passage

$$\alpha_i = \text{softmax}_i \mathbf{q}^T \mathbf{W}_s \tilde{\mathbf{p}}_i$$

$$\mathbf{o} = \sum_i \alpha_i \tilde{\mathbf{p}}_i$$

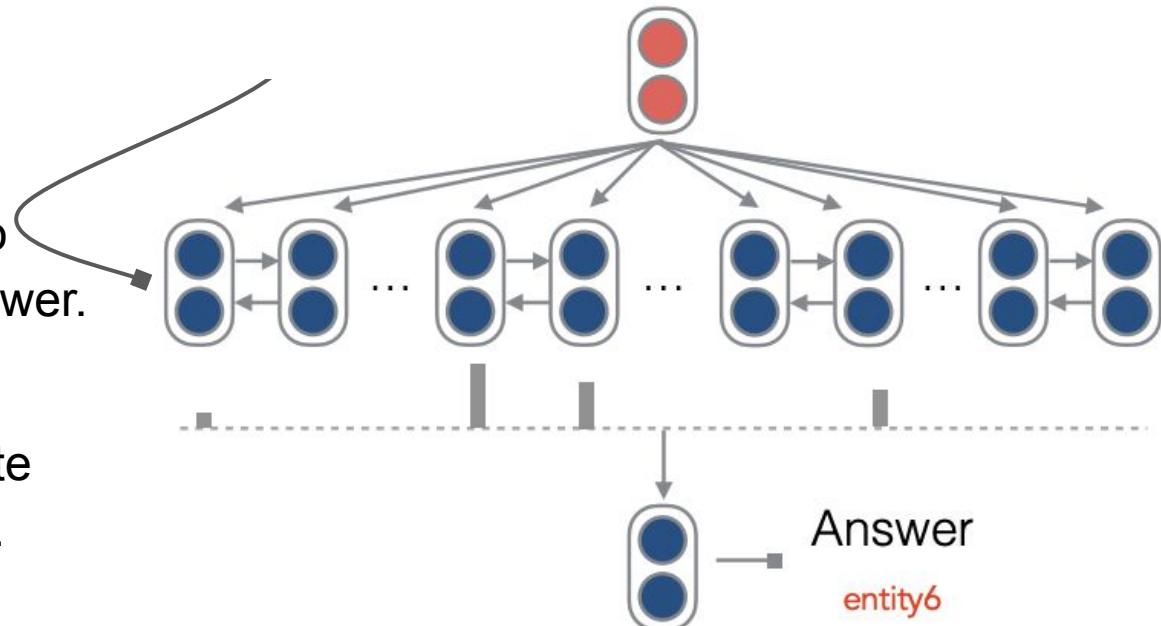
Two attention heads are used to find the *start* and *end* of the answer.

Attention is computed between encoded question and RNN state corresponding to every position.

Potential solutions

Question

characters in " @placeholder " movies have gradually become more diverse



How to make it better

- Use extra information about the text
 - Char embeddings
 - Linguistic features: PoS and NER tags
 - ...

PoS tagging

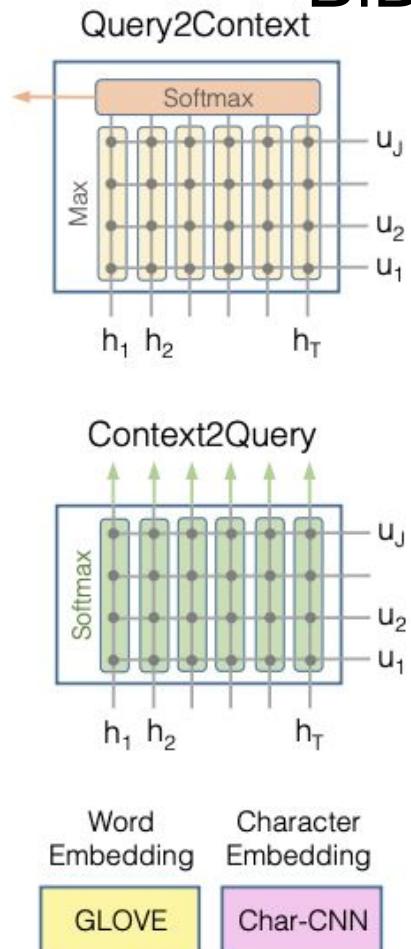
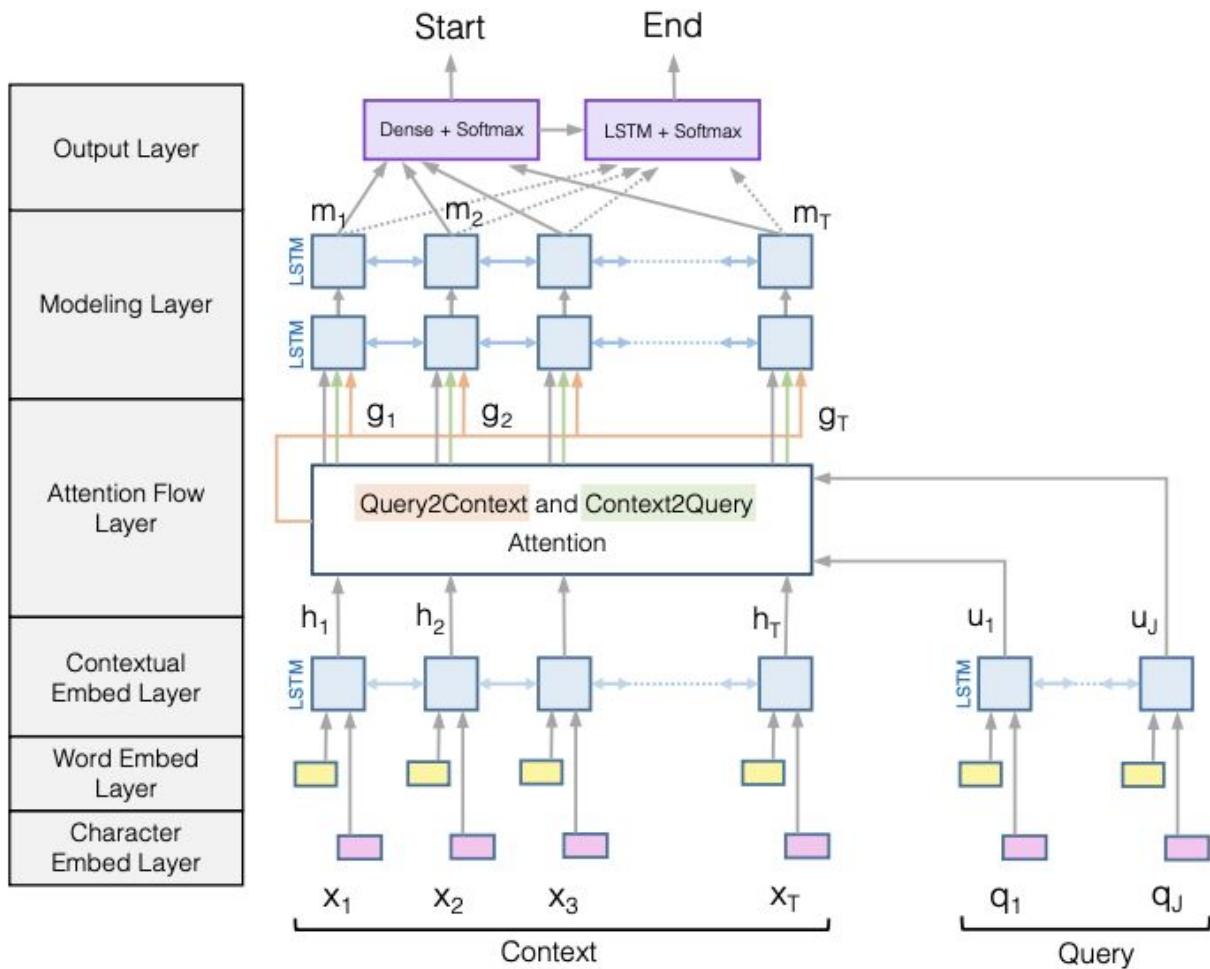
Pred. Tag	Actual Tag	Correct?	Token
PUNCT	PUNCT	✓	[
DET	DET	✓	this
NOUN	NOUN	✓	killing
ADP	ADP	✓	of
DET	DET	✓	a
ADJ	ADJ	✓	respected
NOUN	NOUN	✓	cleric
AUX	AUX	✓	will
AUX	AUX	✓	be
VERB	VERB	✓	causing
PRON	PRON	✓	us
NOUN	NOUN	✓	trouble
ADP	ADP	✓	for
NOUN	NOUN	✓	years
PART	PART	✓	to
VERB	VERB	✓	come
PUNCT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

- PoS tagging can be performed using
 - Rule-based taggers
 - Dynamic programming
 - Models based on CRF (Conditional Random Field)
 - Neural Networks
 - etc.

How to make it better

- Use extra information about the text
 - Char embeddings
 - Linguistic features: PoS and NER tags
 - ...
- Better use of attention

BiDAF



source: [Bidirectional Attention Flow for Machine Comprehension](#)

- There are variants of and improvements to the BiDAF BiDAF architecture, but **the central idea** is the **Attention Flow layer**: attention should flow both ways – from the context to the question and from the question to the context
- Make similarity matrix (with w of dimension 6d):

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

- Context-to-Question (C2Q) attention (which query words are most relevant to each context word):

$$\alpha^i = \text{softmax}(S_{i,:}) \in \mathbb{R}^M \quad \forall i \in \{1, \dots, N\}$$

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

- Attention Flow: BiDAF
 - attention should flow both ways – from the context to the question and from the question to the context
- Question-to-Context (Q2C) attention:
 - the weighted sum of the most important words in the context with respect to the query – slight asymmetry through max

$$\mathbf{m}_i = \max_j S_{ij} \in \mathbb{R} \quad \forall i \in \{1, \dots, N\}$$

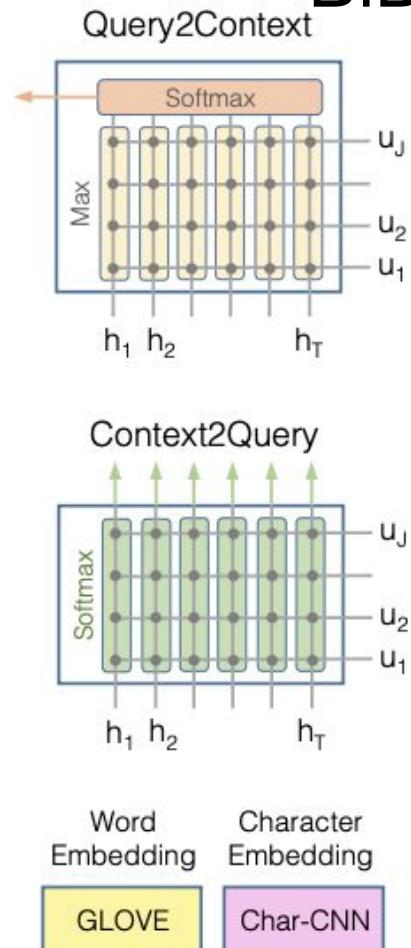
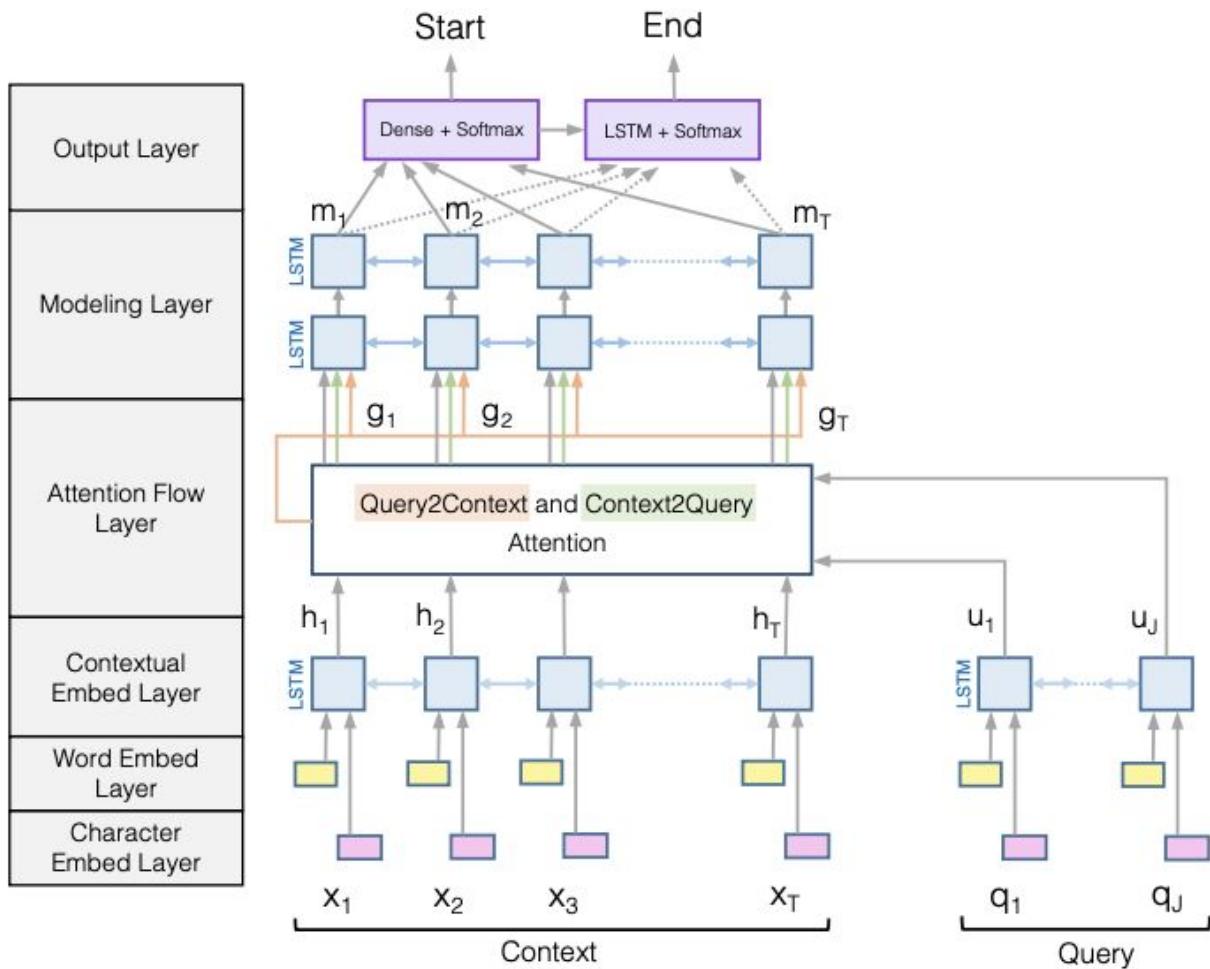
$$\beta = \text{softmax}(\mathbf{m}) \in \mathbb{R}^N$$

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

- For each passage position, output of BiDAF layer is:

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

BiDAF



source: [Bidirectional Attention Flow for Machine Comprehension](#)

Open-Domain Question Answering

Open-domain QA

SQuAD, TREC, WebQuestions, WikiMovies

Q: How many of Warsaw's inhabitants spoke Polish in 1933?



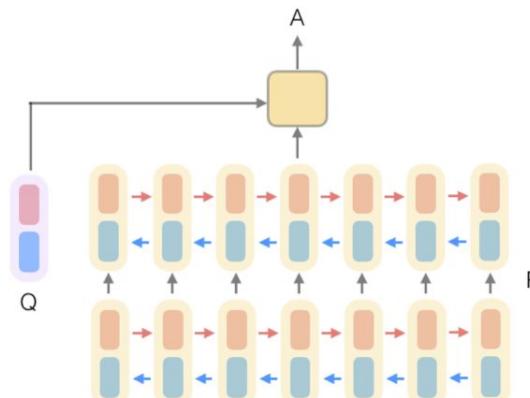
WIKIPEDIA
The Free Encyclopedia

**Document
Retriever**



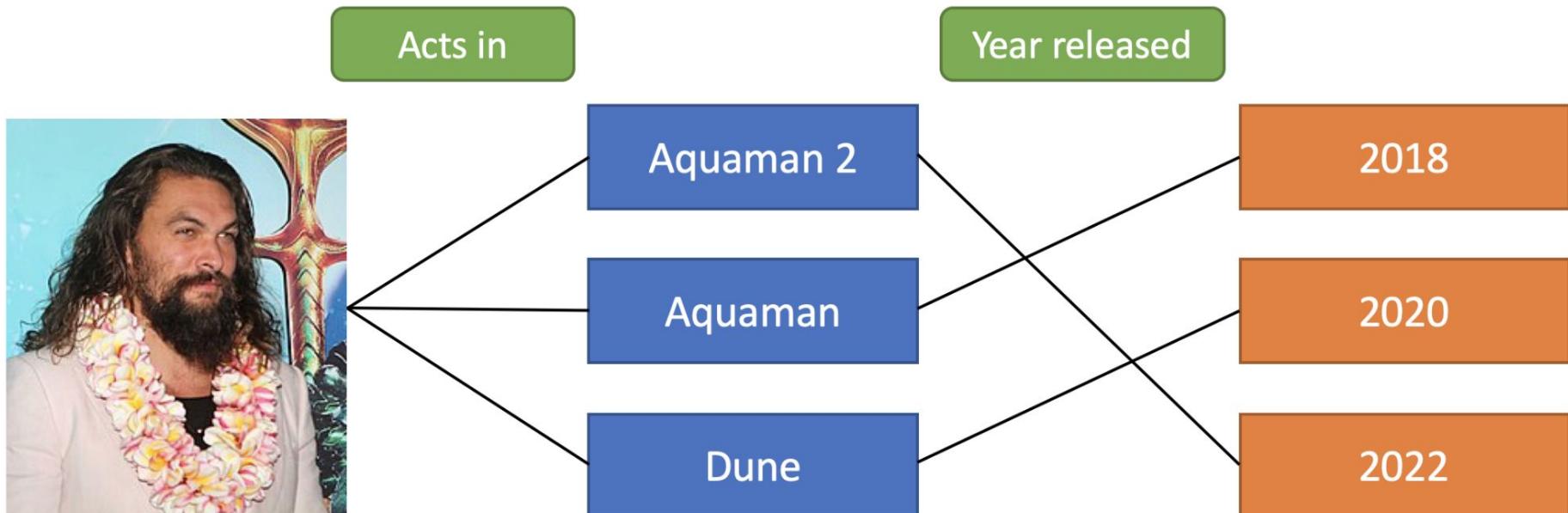
**Document
Reader**

833,500

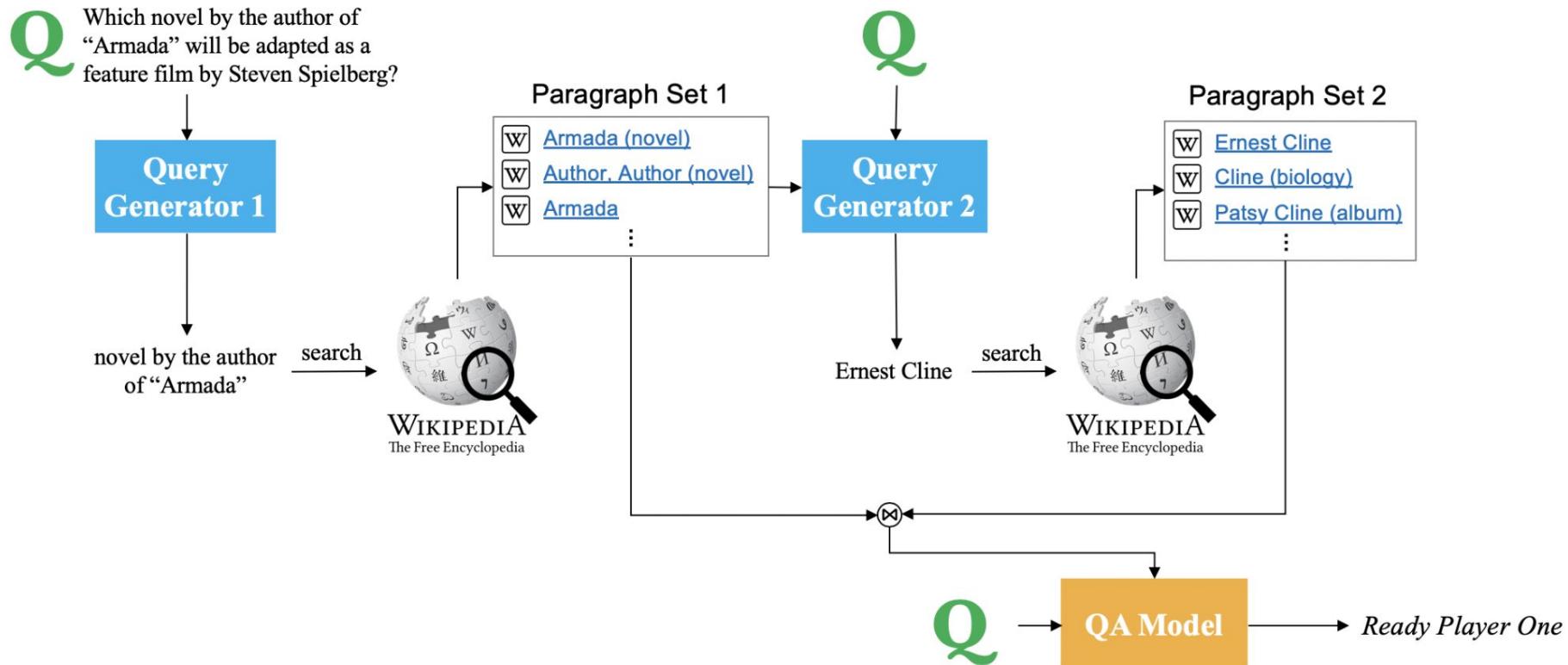


Possible problems

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



Possible solutions

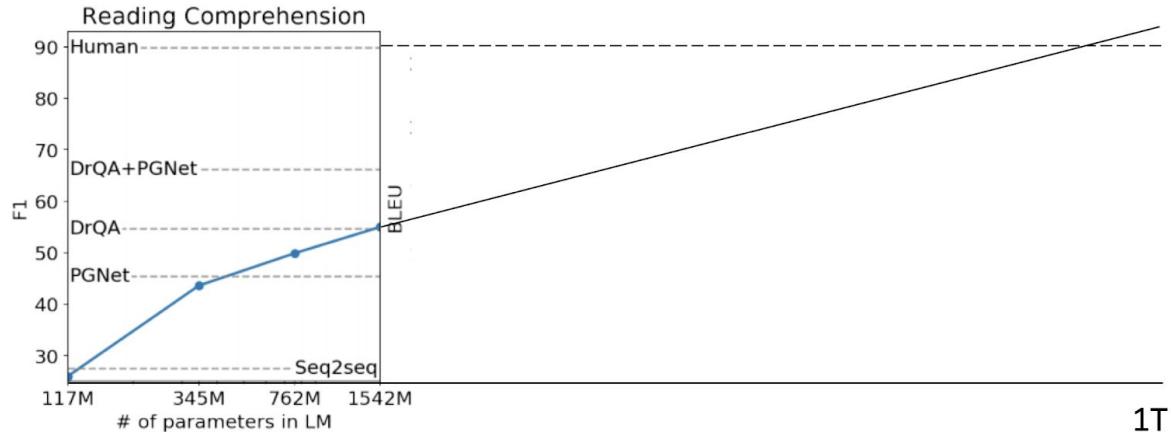
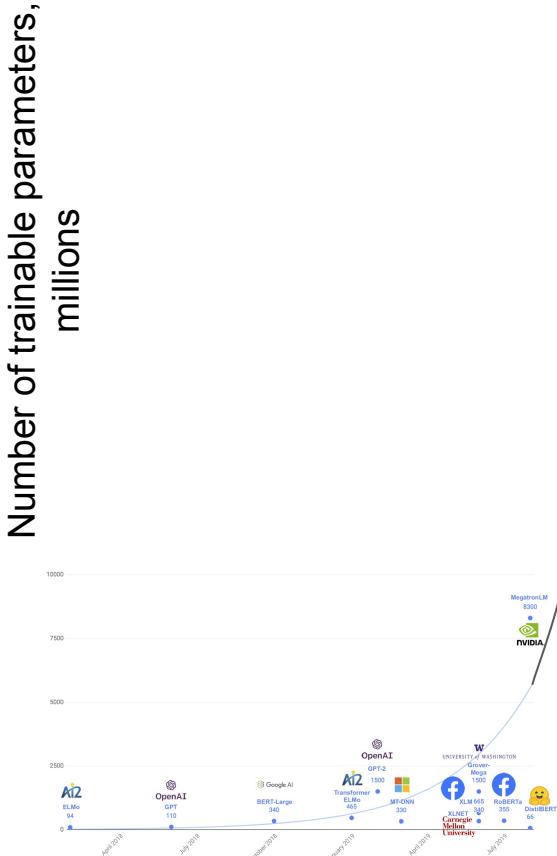


More on GPT

2020 achievements: GPT-3

GPT-3, May 2020

Proportions are not preserved for visual sake



Hypothesis from Stanford CS224N Lecture 20 (2019)

Reaction: GPT-3



Sam Altman  @sama · Jul 19

The GPT-3 hype is way too much. It's impressive (thanks for the nice compliments!) but it still has serious weaknesses and sometimes makes very silly mistakes. AI is going to change the world, but GPT-3 is just a very early glimpse. We have a lot still to figure out.

148 963 7K 



Paul Graham  @paulg · Jul 19

Hackers are fascinated by GPT-3. To everyone else it seems a toy.

Pattern seem familiar to anyone?

153 604 5K 

Reaction: GPT-3



Andriy Burkov • Following

ML at Gartner, author of The Hundred-Page Machine Learning Book

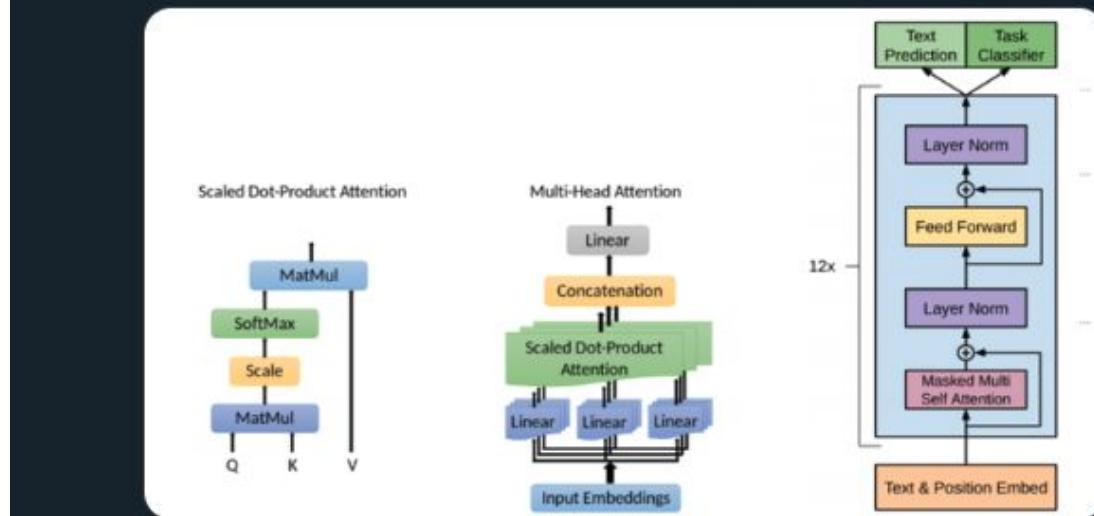
2d • 56

GPT-3 is the closest thing to artificial general intelligence (AGI) that I ever saw.
It's so strong that it makes me nervous.



Andrej Karpathy ✅ @karpathy · Jul 19

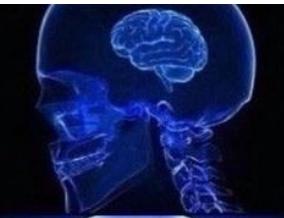
The transformer architecture of GPT upper bounds its ability at memorization. It cannot learn many algorithms due to the functional form of its forward pass, and spends a fixed compute per token - i.e. it can't "think for a while". Progress here critical, likely but non-trivial.



Source: LinkedIn, Twitter

Reaction: GPT-3

**PRE-SOFTWARE:
SPECIAL-PURPOSE
COMPUTER**



**SOFTWARE 1.0:
DESIGN
THE ALGORITHM**



**SOFTWARE 2.0:
DESIGN
THE DATASET**



**SOFTWARE 3.0:
DESIGN
THE PROMPT**



Andrej Karpathy
@karpathy

Replies to @ch402

Love the idea for Software 3.0 😂. Programming moving from curating datasets to curating prompts to make the meta learner "get" the task it's supposed to be doing. LOL

- Great way to understand GPT
 - ~300 lines of code and examples

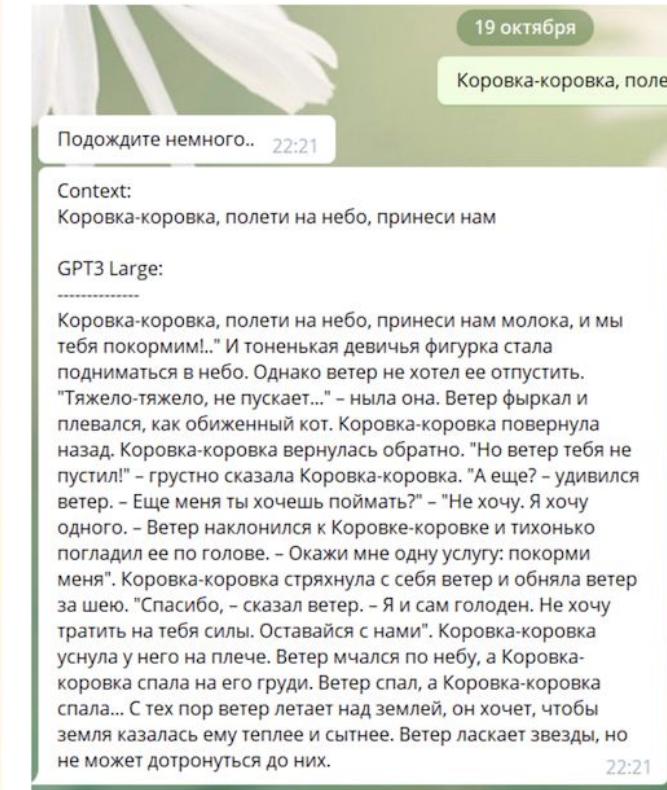
available GPT implementations



minGPT



GPT-3 Large for Russian Language



source: [Сбер выложил русскоязычную модель GPT-3 Large с 760 миллионами параметров в открытый доступ](#)



- Question answering systems guide us one step closer to human-like NLP systems
- Refer to the original papers on [Transformer-XL](#), [BiDAF](#) and [SQuAD](#), there are a lot of interesting ideas in there
- For Russian language:
 - [SberQuAD paper](#) provides a great aggregation of available materials, libraries and pretrained models
 - [GPT-3 trained on Russian texts](#)
 - [deeppavlov.ai](#) and [Natasha project](#) provide many useful materials and pretrained models