Long-Context Language Models

By Charles Alba

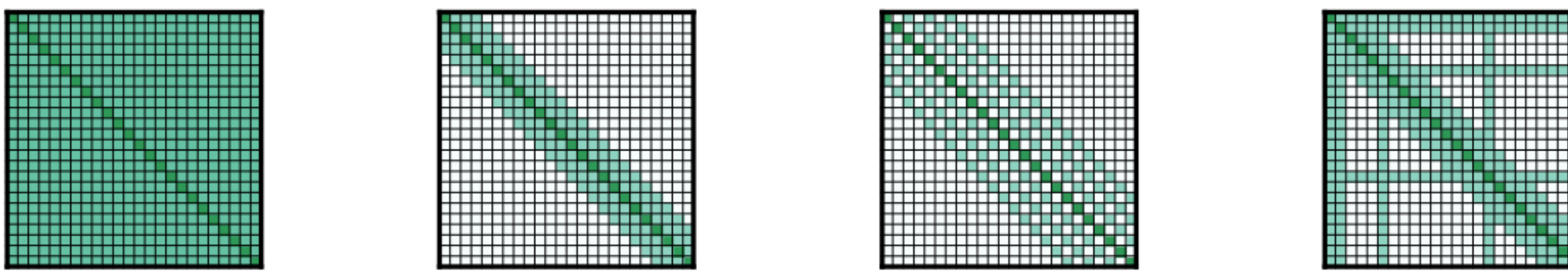
Washington University in St. Louis

Agenda

- How can pre-trained language models process long documents:
  - Longformer: The long-document Transformer
    by *Beltagy, Peters, and Cohan*
- Are LLMs effective in ‘digesting’ long contexts?
  - Lost in the Middle: How Language Models Use Long Contexts
    by *Liu et al*
- Q&A

Longformer: The Long-Document Transformer

Iz Beltagy* **Matthew E. Peters*** **Arman Cohan***
Allen Institute for Artificial Intelligence, Seattle, WA, USA
{beltagy,matthewp,armanc}@allenai.org



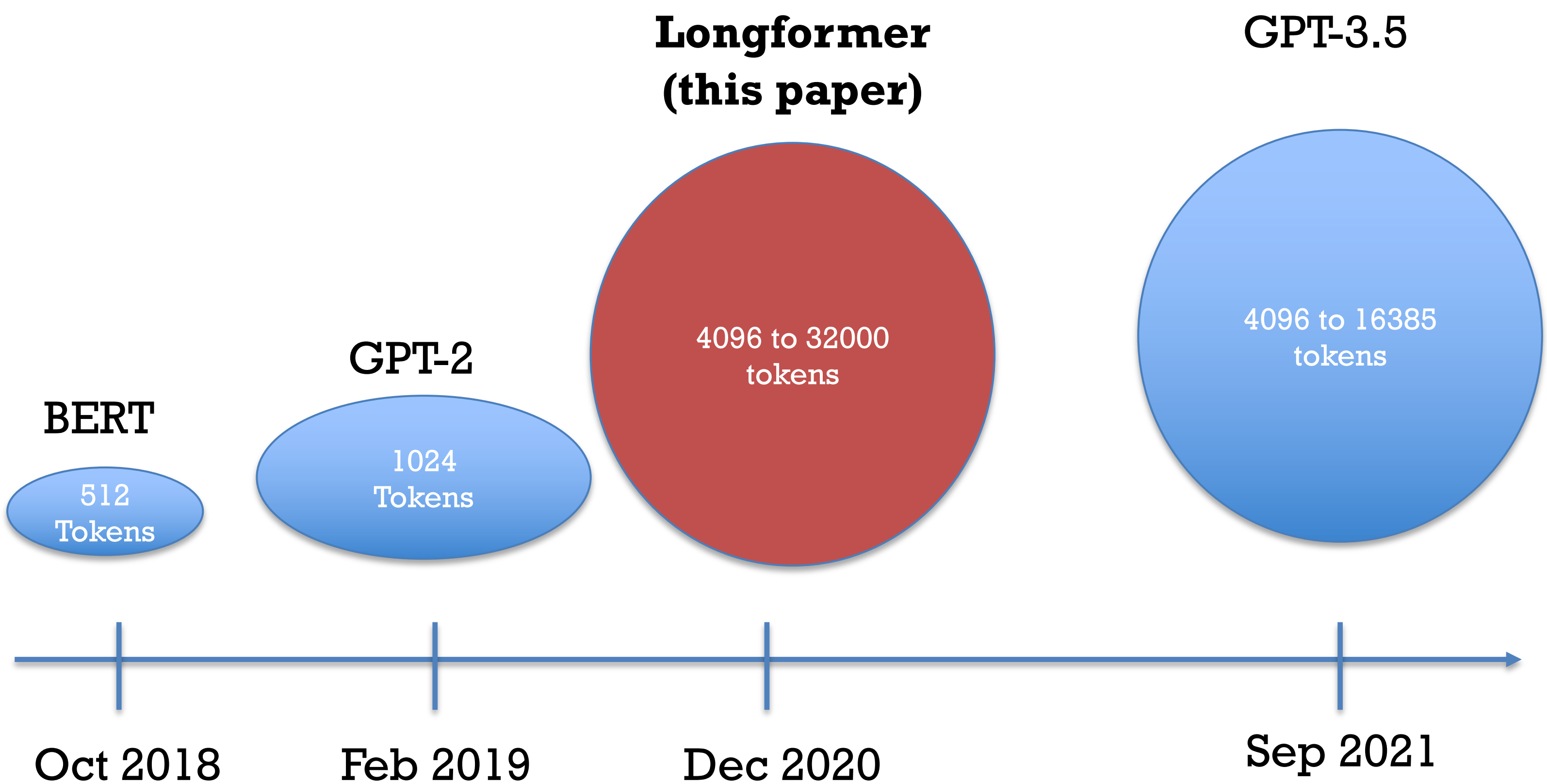
(a) Full n^2 attention (b) Sliding window attention (c) Dilated sliding window (d) Global+sliding window

Let's put things into context:

What's so great about Longformer when SOTA models can contextualize up to 32k tokens?

MODEL	DESCRIPTION	CONTEXT WINDOW	TRAINING DATA
gpt-3.5-turbo-0125	New **Updated GPT 3.5 Turbo** The latest GPT-3.5 Turbo model with higher accuracy at responding in requested formats and a fix for a bug which caused a text encoding issue for non-English language function calls. Returns a maximum of 4,096 output tokens. Learn more.	16,385 tokens	Up to Sep 2021
gpt-3.5-turbo	Currently points to `gpt-3.5-turbo-0125`.	16,385 tokens	Up to Sep 2021
gpt-3.5-turbo-1106	GPT-3.5 Turbo model with improved instruction following, JSON mode, reproducible outputs, parallel function calling, and more. Returns a maximum of 4,096 output tokens. Learn more.	16,385 tokens	Up to Sep 2021
gpt-3.5-turbo-instruct	Similar capabilities as GPT-3 era models. Compatible with legacy Completions endpoint and not Chat Completions.	4,096 tokens	Up to Sep 2021
gpt-3.5-turbo-16k	Legacy Currently points to `gpt-3.5-turbo-16k-0613`.	16,385 tokens	Up to Sep 2021
gpt-3.5-turbo-0613	Legacy Snapshot of `gpt-3.5-turbo` from June 13th 2023. Will be deprecated on June 13, 2024.	4,096 tokens	Up to Sep 2021
gpt-3.5-turbo-16k-0613	Legacy Snapshot of `gpt-3.5-16k-turbo` from June 13th 2023. Will be deprecated on June 13, 2024.	16,385 tokens	Up to Sep 2021

This paper was well ahead of its time!



How did we deal with scenarios where the text exceeds the max number of tokens?

Method 1: Truncation

Transformer-based models are unable to process long sequences due to their self-attention operation, which scales quadratically with the sequence length. To address this limitation, we introduce the Longformer with an attention mechanism that scales linearly with sequence length, making it easy to process documents of thousands of tokens or longer. Longformer's attention mechanism is a drop-in replacement for the standard self-attention and combines a local windowed attention with a task motivated global attention. ~~Following prior work on long sequence transformers, we evaluate Longformer on character level language modeling and achieve state of the art results on text8 and enwik8. In contrast to most prior work, we also pretrain Longformer and finetune it on a variety of downstream tasks. Our pretrained Longformer consistently outperforms RoBERTa on long document tasks and sets new state of the art results on WikiHop and TriviaQA. We finally introduce the Longformer-Encoder-Decoder (LED), a Longformer variant for supporting long document generative sequence-to-sequence tasks, and demonstrate its effectiveness on the arXiv summarization dataset.[1]~~

How did we deal with scenarios where the text exceeds the max number of tokens?

Method 2: Divide them into chunks

Transformer-based models are unable to process long sequences due to their self-attention operation, which scales quadratically with the sequence length. To address this limitation, we introduce the Longformer with an attention mechanism that scales linearly with sequence length, making it easy to process documents of thousands of tokens or longer. Longformer's attention mechanism is a drop-in replacement for the standard self-attention and combines

Chunk #1

a local windowed attention with a task motivated global attention. Following prior work on long-sequence transformers, we evaluate Longformer on character-level language modeling and achieve state-of-the-art results on `text8` and `enwik8`. In contrast to most prior work, we also pretrain Longformer and finetune it on a variety of downstream tasks. Our pretrained Longformer consistently outperforms RoBERTa on long document tasks

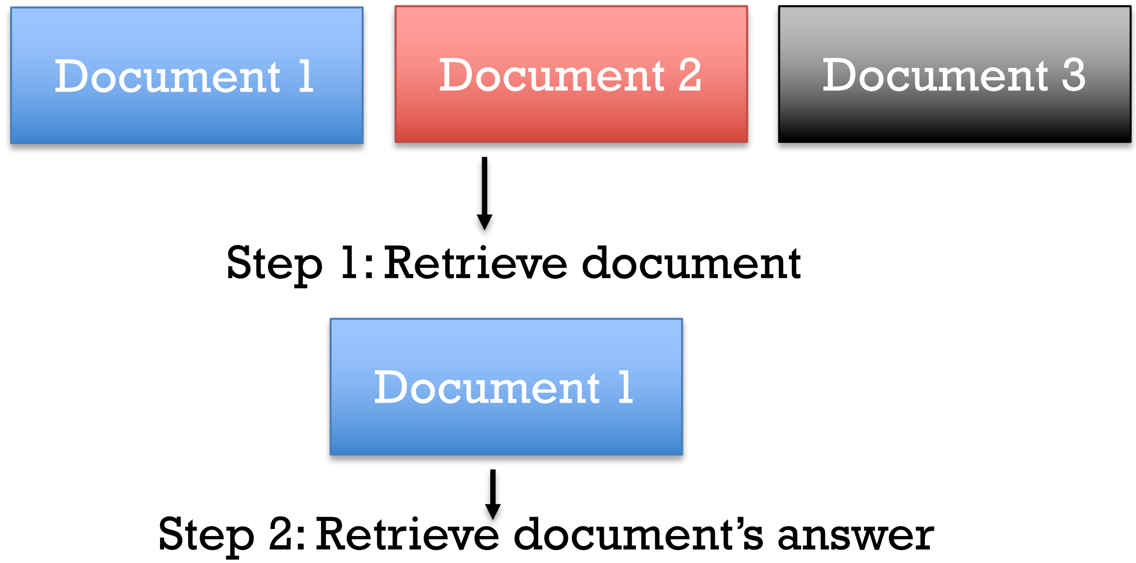
Chunk #2

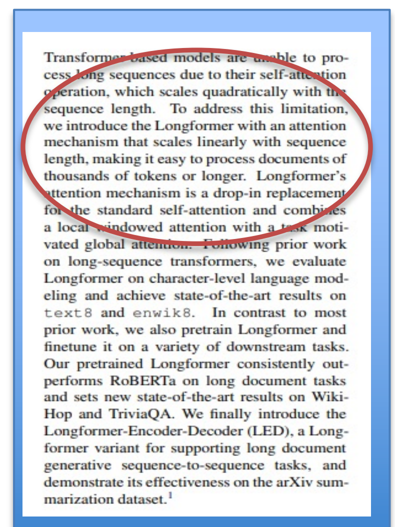
and sets new state-of-the-art results on WikiHop and TriviaQA. We finally introduce the Longformer-Encoder-Decoder (LED), a Longformer variant for supporting long document generative sequence-to-sequence tasks, and demonstrate its effectiveness on the arXiv summarization dataset.[1]

Chunk #3

How did we deal with scenarios where the text exceeds the max number of tokens?

Method 3: Two-stage extraction

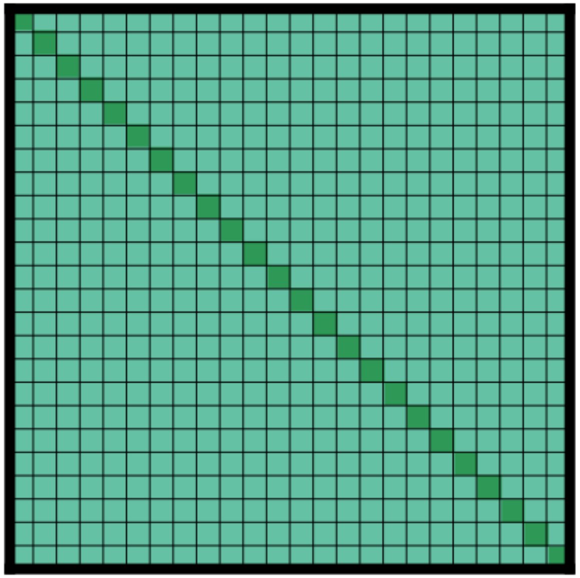




Transformer-based models are unable to process long sequences due to their self-attention operation, which scales quadratically with the sequence length. To address this limitation, we introduce the Longformer with an attention mechanism that scales linearly with sequence length, making it easy to process documents of thousands of tokens or longer. Longformer's attention mechanism is a drop-in replacement for the standard self-attention and combines a local windowed attention with a task motivated global attention. Following prior work on long-sequence transformers, we evaluate Longformer on character-level language modeling and achieve state-of-the-art results on text8 and enwik8. In contrast to most prior work, we also pretrain Longformer and finetune it on a variety of downstream tasks. Our pretrained Longformer consistently outperforms RoBERTa on long document tasks and sets new state-of-the-art results on WikiHop and TriviaQA. We finally introduce the Longformer-Encoder-Decoder (LED), a Longformer variant for supporting long document generative sequence-to-sequence tasks, and demonstrate its effectiveness on the arXiv summarization dataset.[1]

We suffer from Information Loss

How does the 'traditional' self-attention mechanism work?

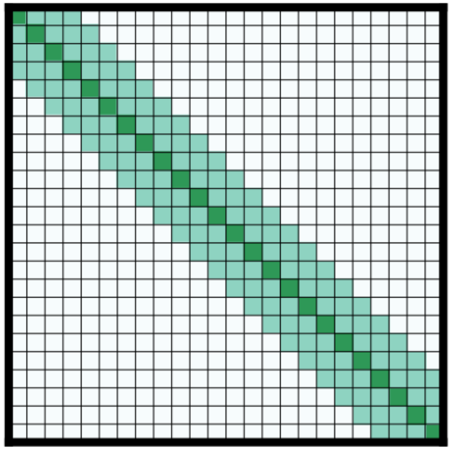


(a) Full n^2 attention

*I **love** Washington University and it is a great school!*

- All words are attended to!
- $O(n^2)$

Proposed sliding window attention

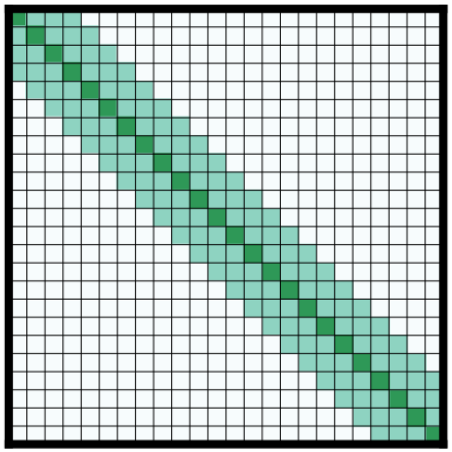


(b) Sliding window attention

Round 1:

I love Washington University and it is a great school!

Proposed sliding window attention

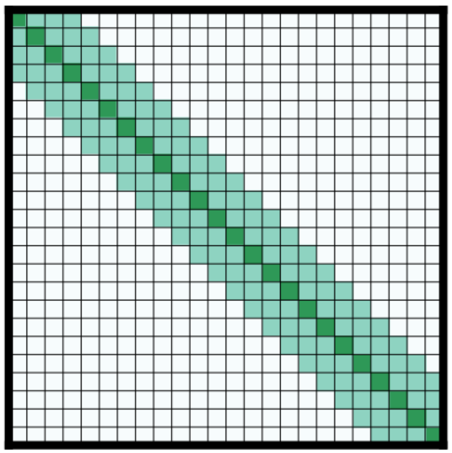


(b) Sliding window attention

<u>Round 2:</u>

*I love **<u>Washington</u>** University and it is a great school!*

Proposed sliding window attention

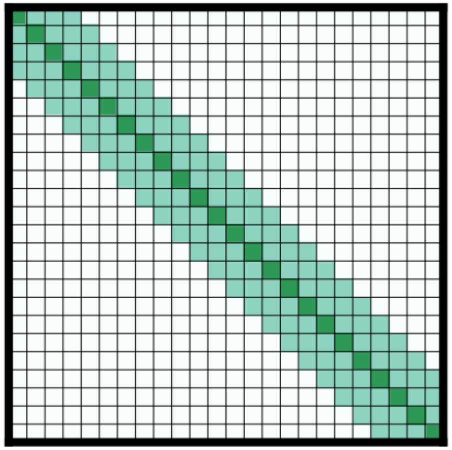


(b) Sliding window attention

<u>Round n-1:</u>

I love Washington University and it is a great school!

Proposed sliding window attention



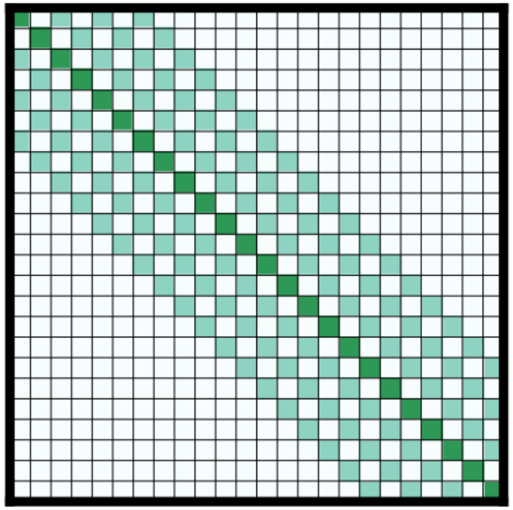
(b) Sliding window attention

Round n-1:

I love Washington University and it is a great school!

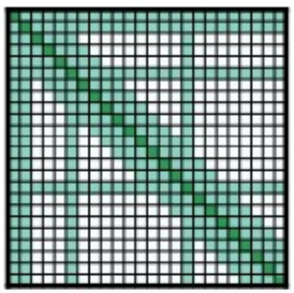
- Similar to classic CNNs!
- O(w*n)
  where w is sliding window size

Proposed dilated sliding window attention



(c) Dilated sliding window

Proposed global + sliding window



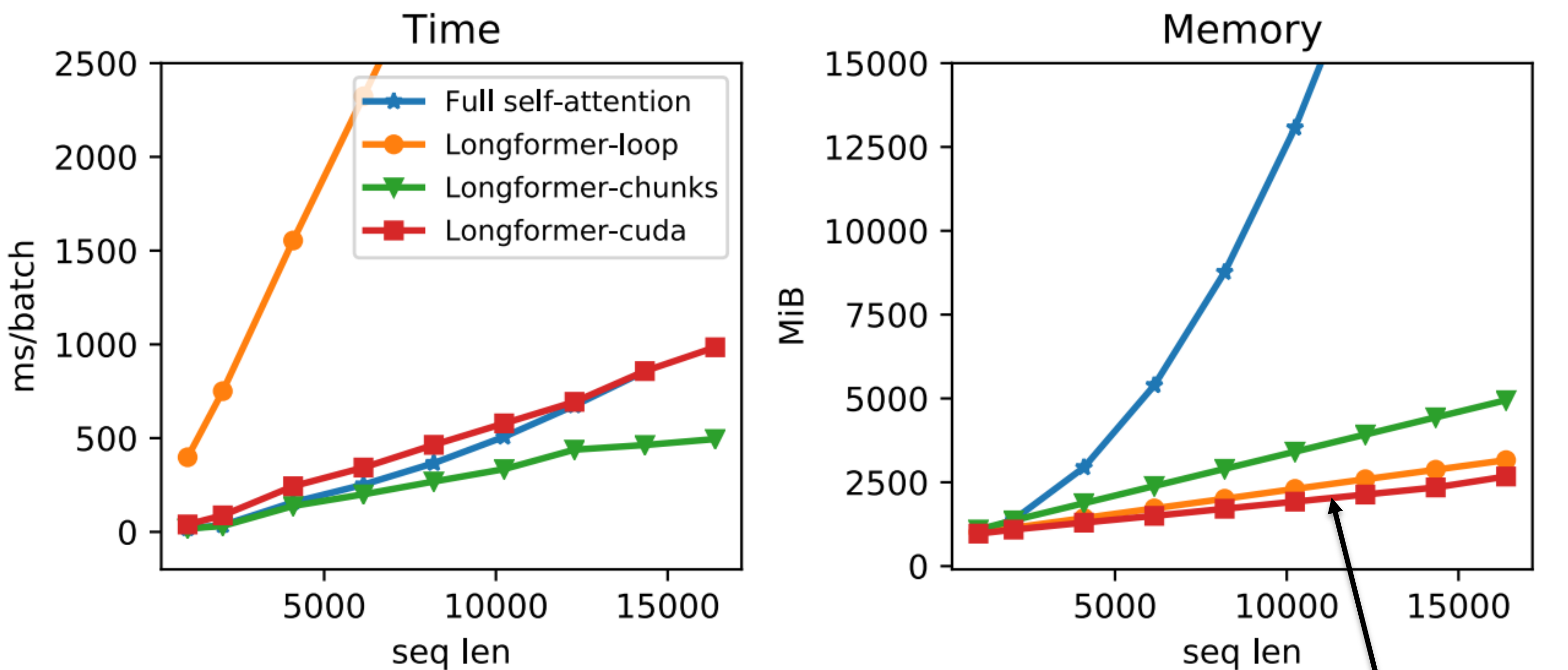
(d) Global+sliding window

[CLS] I love Washington University and it is a great school !

Preselected!

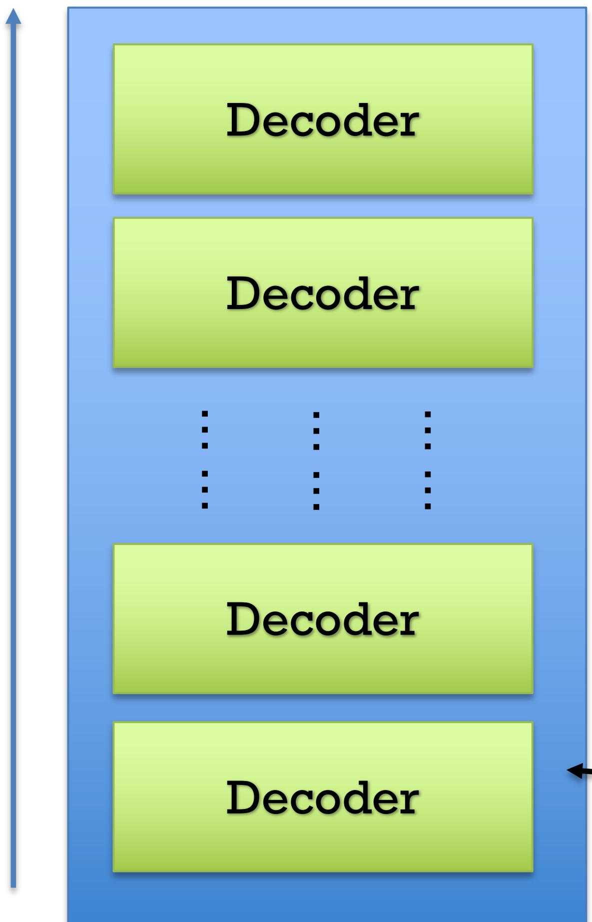
Note: Still O(n)!

Effect of sliding window on memory consumption



Linear increase in memory consumption!!!

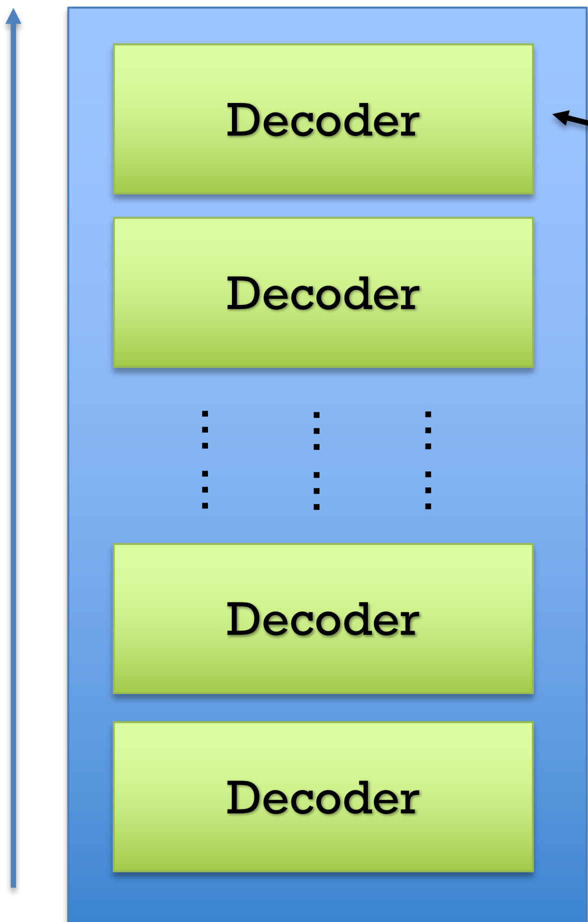
Can this be implemented with Autoregressive Language Modeling?



Note: we use a sliding window attention here!

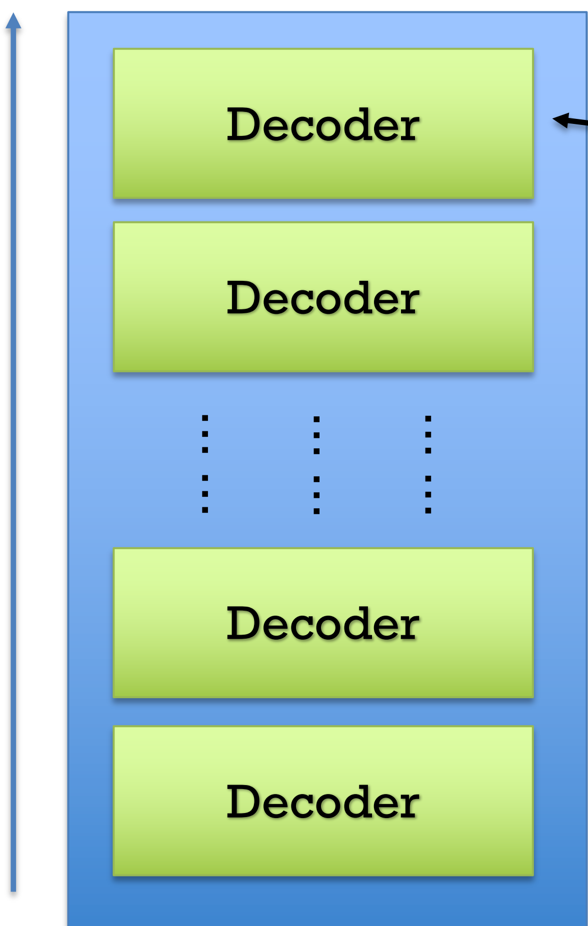
Start with a **small** window
Size of w=32

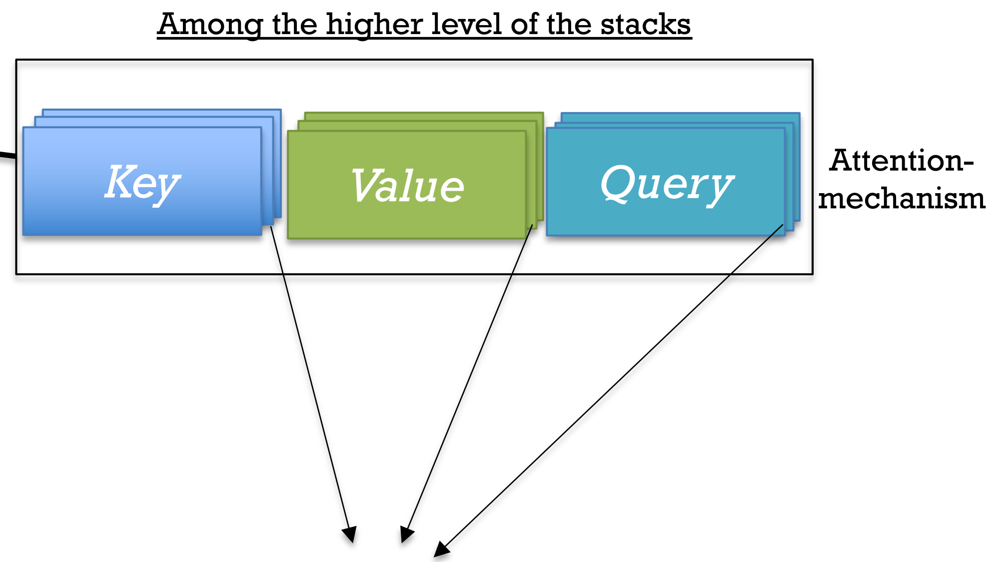
Can this be implemented with Autoregressive Language Modeling?



Increase to a **<u>larger</u>** window size in the higher stacks (w=512)

Can this be implemented with Autoregressive Language Modeling?





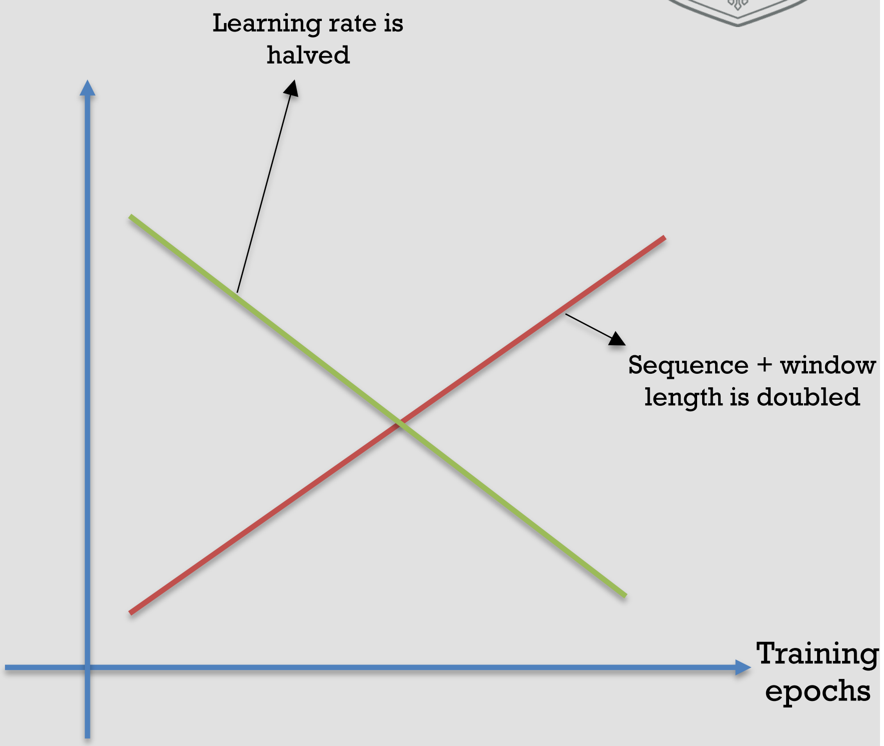
- Select 2 layers to perform dilation sliding window!
- The other layers still use sliding window attention

Training

To speed up training, we

- Double attention size and window lengths
- Half learning rate

Across the training stages



Results

↓ is better

Model	#Param	Dev	Test
Dataset `text8`			
T12 (Al-Rfou et al., 2018)	44M	-	1.18
Adaptive (Sukhbaatar et al., 2019)	38M	1.05	1.11
BP-Transformer (Ye et al., 2019)	39M	-	1.11
Our Longformer	41M	1.04	**1.10**
Dataset `enwik8`			
T12 (Al-Rfou et al., 2018)	44M	-	1.11
Transformer-XL (Dai et al., 2019)	41M	-	1.06
Reformer (Kitaev et al., 2020)	-	-	1.05
Adaptive (Sukhbaatar et al., 2019)	39M	1.04	1.02
BP-Transformer (Ye et al., 2019)	38M	-	1.02
Our Longformer	41M	1.02	**1.00**

Table 2: *Small* model BPC on `text8` & `enwik8`

Model	#Param	Test BPC
Transformer-XL (18 layers)	88M	1.03
Sparse (Child et al., 2019)	≈100M	0.99
Transformer-XL (24 layers)	277M	0.99
Adaptive (Sukhbaatar et al., 2019)	209M	0.98
Compressive (Rae et al., 2020)	277M	0.97
Routing (Roy et al., 2020)	≈223M	0.99
Our Longformer	102M	0.99

Table 3: Performance of *large* models on `enwik8`

Ablation studies

Model	Dev BPC
Decreasing w (from 512 to 32)	1.24
Fixed w (= 230)	1.23
Increasing w (from 32 to 512)	**1.21**
No Dilation	1.21
Dilation on 2 heads	**1.20**

Pretraining and fine-tuning?

Pretrained from RoBERTa then finetuned on 3 tasks:

Question Answering*

Coreference resolution

Document classification*

* Global attention is used on these tasks

Achieves amazing performances on multiple tasks

	QA			Coref.	Classification	
Model	WikiHop	TriviaQA	HotpotQA	OntoNotes	IMDB	Hyperpartisan
RoBERTa-base	72.4	74.3	63.5	78.4	95.3	87.4
Longformer-base	**75.0**	**75.2**	**64.4**	**78.6**	**95.7**	**94.8**

Table 7: Summary of finetuning results on QA, coreference resolution, and document classification. Results are on the development sets comparing our Longformer-base with RoBERTa-base. TriviaQA, Hyperpartisan metrics are F1, WikiHop and IMDB use accuracy, HotpotQA is joint F1, OntoNotes is average F1.

Encoder-Decoder model?

Results

	R-1	R-2	R-L
Discourse-aware (2018)	35.80	11.05	31.80
Extr-Abst-TLM (2020)	41.62	14.69	38.03
Dancer (2020)	42.70	16.54	38.44
Pegasus (2020)	44.21	16.95	38.83
LED-large (seqlen: 4,096) (ours)	44.40	17.94	39.76
BigBird (seqlen: 4,096) (2020)	**46.63**	19.02	41.77
LED-large (seqlen: 16,384) (ours)	**46.63**	**19.62**	**41.83**

Table 11: Summarization results of Longformer-Encoder-Decoder (LED) on the arXiv dataset. Metrics from left to right are ROUGE-1, ROUGE-2 and ROUGE-L.

Works particularly well with longer input sizes!!!

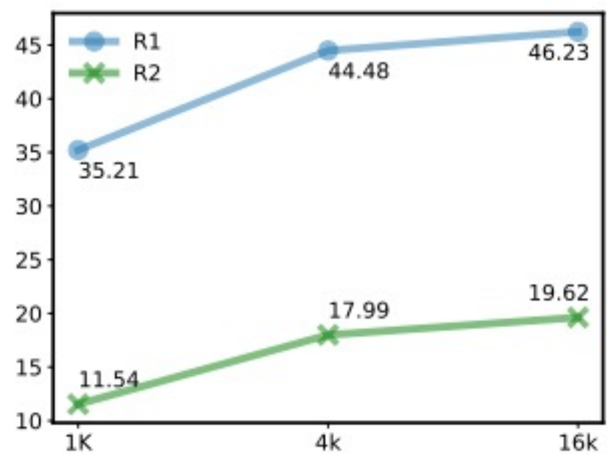


Figure 3: ROUGE-1 and ROUGE-2 of LED when varying the input size (arXiv validation set).

Does input context length affect performance?

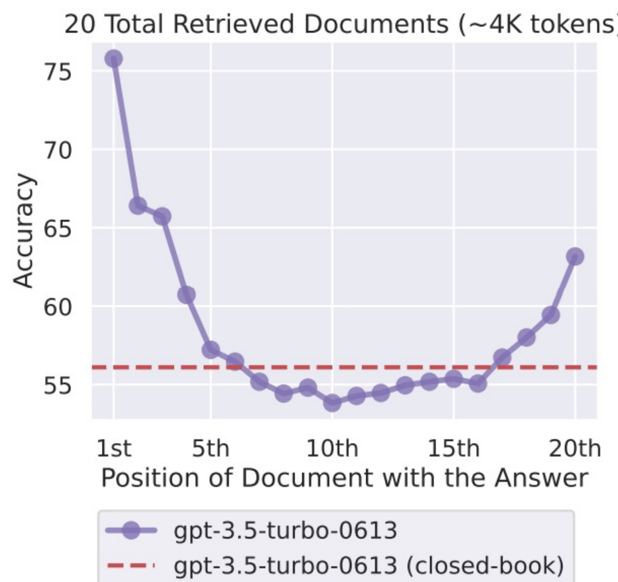
Does the position of the relevant information affect performance?

We demonstrated how LLMs can gain contextual understanding from long documents!!!

Exactly how well can models reason over long contexts?

Lost in the Middle: How Language Models Use Long Contexts

Nelson F. Liu[1*] **Kevin Lin**[2] **John Hewitt**[1] **Ashwin Paranjape**[3]
Michele Bevilacqua[3] **Fabio Petroni**[3] **Percy Liang**[1]
[1]Stanford University [2]University of California, Berkeley [3]Samaya AI
nfliu@cs.stanford.edu



Experiment (original input)

Input Context

```
Write a high-quality answer for the given question using only the provided search
results (some of which might be irrelevant).

Document [1](Title: Asian Americans in science and technology) Prize in physics for
discovery of the subatomic particle J/ψ. Subrahmanyan Chandrasekhar shared...
Document [2](Title: List of Nobel laureates in Physics) The first Nobel Prize in
Physics was awarded in 1901 to Wilhelm Conrad Röntgen, of Germany, who received...
Document [3](Title: Scientist) and pursued through a unique method, was essentially
in place. Ramón y Cajal won the Nobel Prize in 1906 for his remarkable...

Question: who got the first nobel prize in physics
Answer:
```

Desired Answer

```
Wilhelm Conrad Röntgen
```

Part I: Experimenting the effect of position on performance

Input Context

```
Write a high-quality answer for the given question using only the provided search
results (some of which might be irrelevant).

Document [1](Title: Asian Americans in science and technology) Prize in physics for
discovery of the subatomic particle J/ψ. Subrahmanyan Chandrasekhar shared...
Document [2](Title: List of Nobel laureates in Physics) The first Nobel Prize in
Physics was awarded in 1901 to Wilhelm Conrad Röntgen, of Germany, who received...
Document [3](Title: Scientist) and pursued through a unique method, was essentially
in place. Ramón y Cajal won the Nobel Prize in 1906 for his remarkable...

Question: who got the first nobel prize in physics
Answer:
```

Desired Answer

```
Wilhelm Conrad Röntgen
```

Part I: Experimenting the effect of position on performance

Input Context

```
Write a high-quality answer for the given question
using only the provided search results (some of
which might be irrelevant).

Document [1](Title: List of Nobel laureates in
Physics) ...
Document [2](Title: Asian Americans in science and
technology) ...
Document [3](Title: Scientist) ...

Question: who got the first nobel prize in physics
Answer:
```

Desired Answer

```
Wilhelm Conrad Röntgen
```

Part II: Experimenting the effect of input context length on performance

Input Context

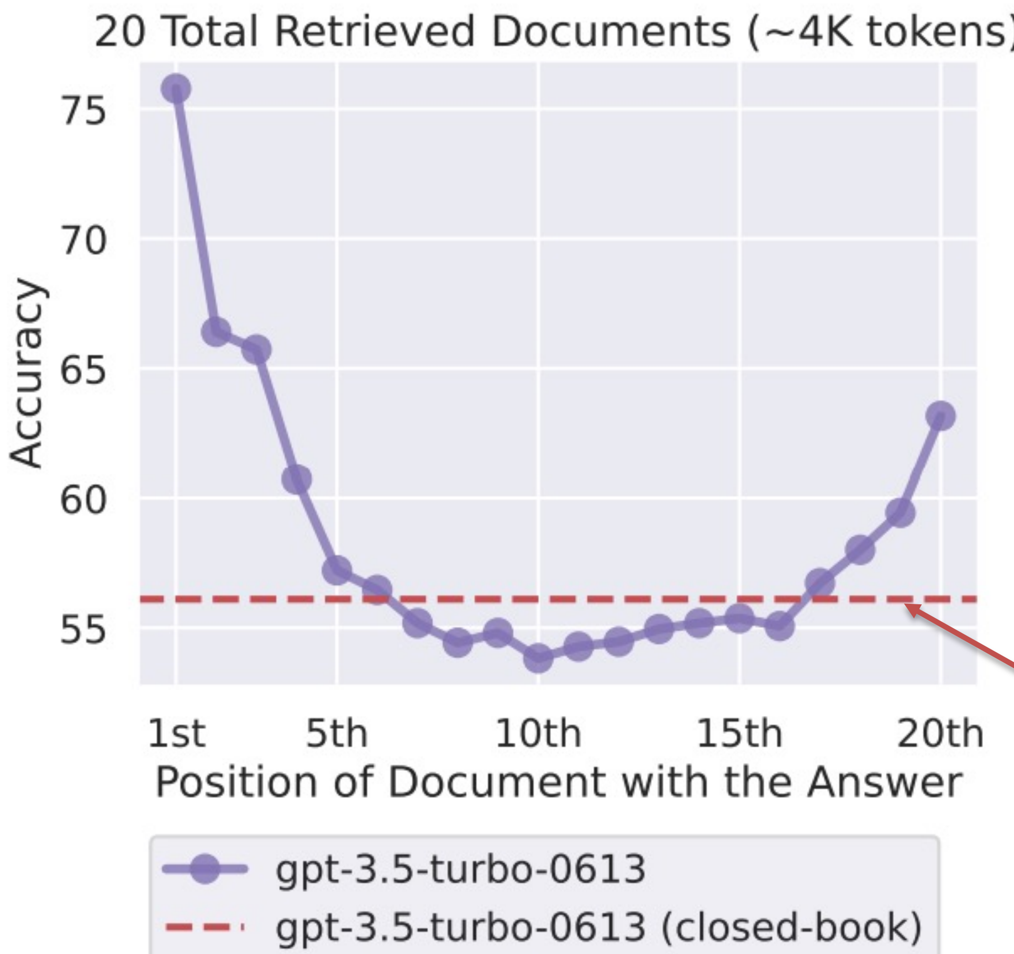
```
Write a high-quality answer for the given question
using only the provided search results (some of
which might be irrelevant).

Document [1](Title: Asian Americans in science and
technology) ...
Document [2](Title: List of Nobel laureates in
Physics) ...
Document [3](Title: Scientist) ...
Document [4](Title: Norwegian Americans) ...
Document [5](Title: Maria Goeppert Mayer) ...

Question: who got the first nobel prize in physics
Answer:
```

Added into the inputs!!!

Desired Answer

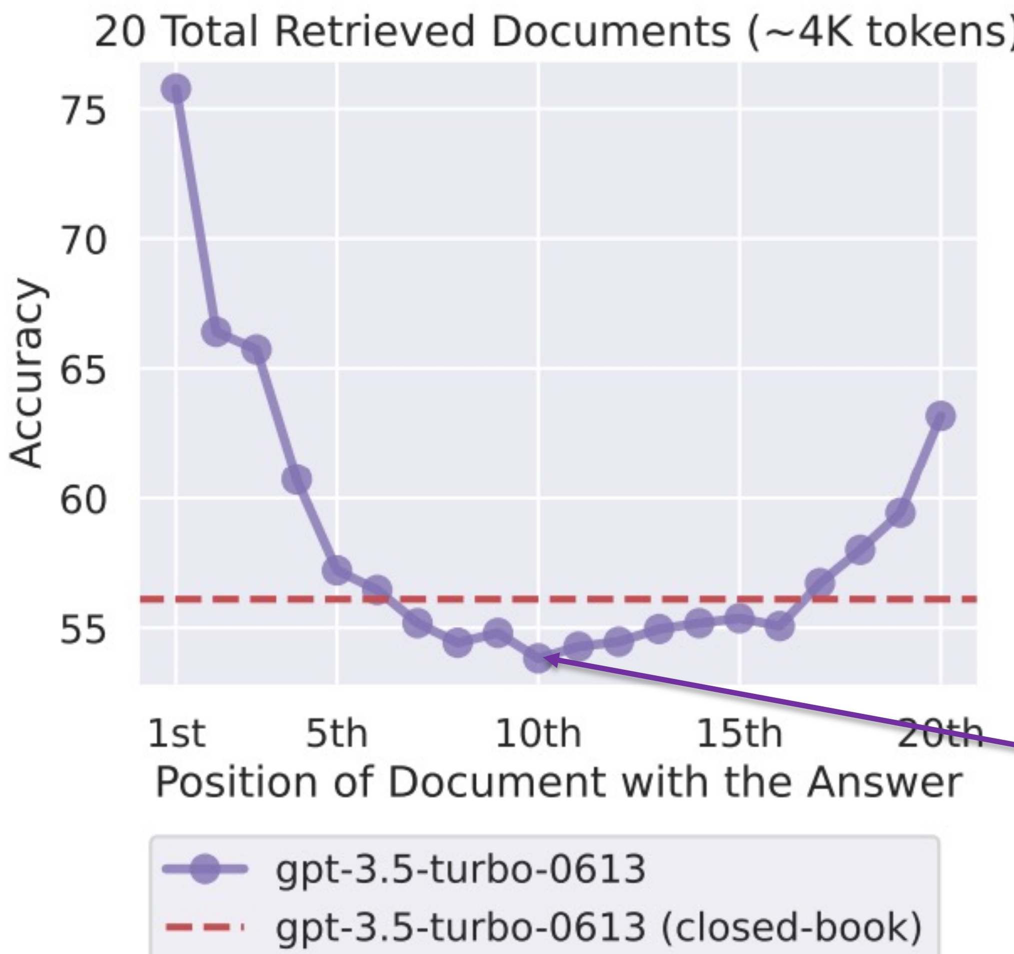
```
Wilhelm Conrad Röntgen
```

We get a 'U' shape!!!



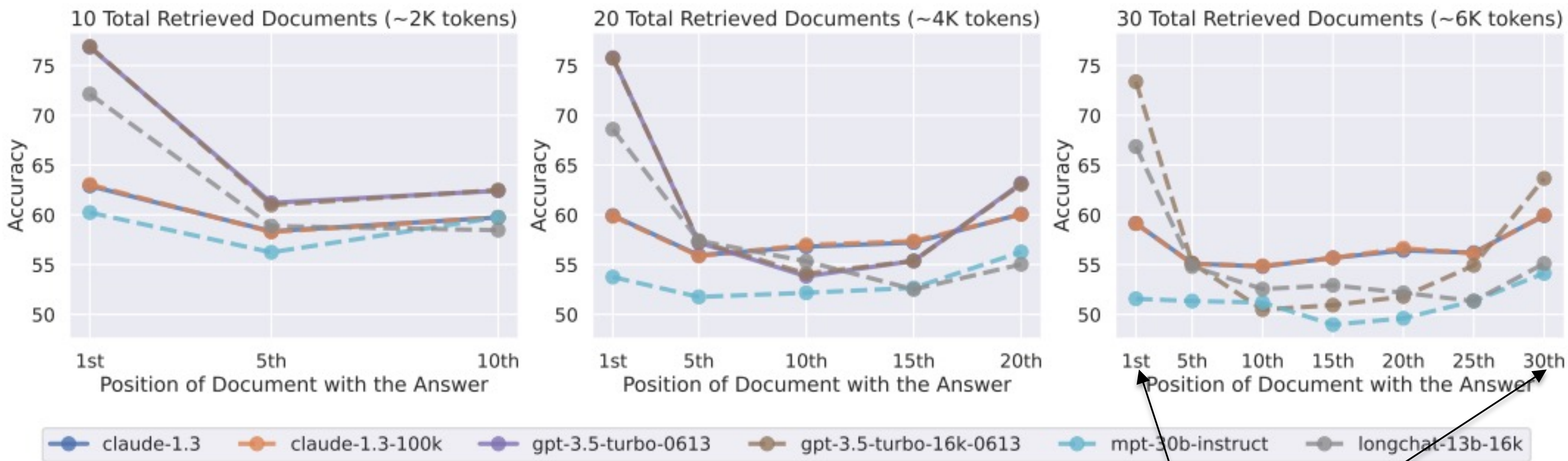
Performance when no documents are provided!!!!

We get a ‘U’ shape!!!



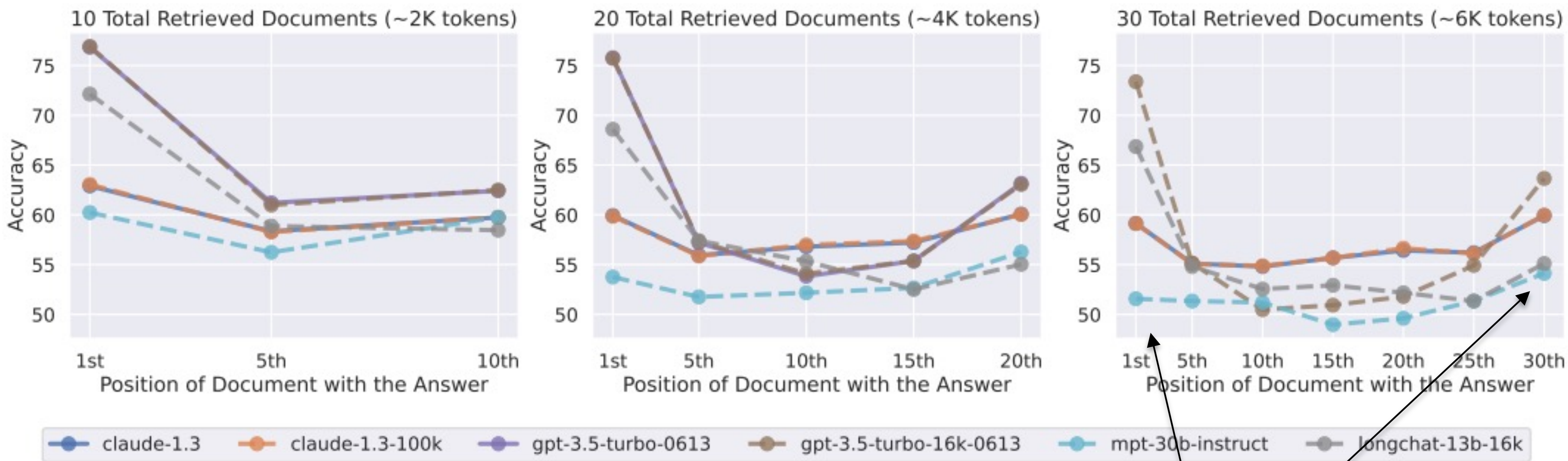
Performs worse!!!

Effect of position and context length on model performance



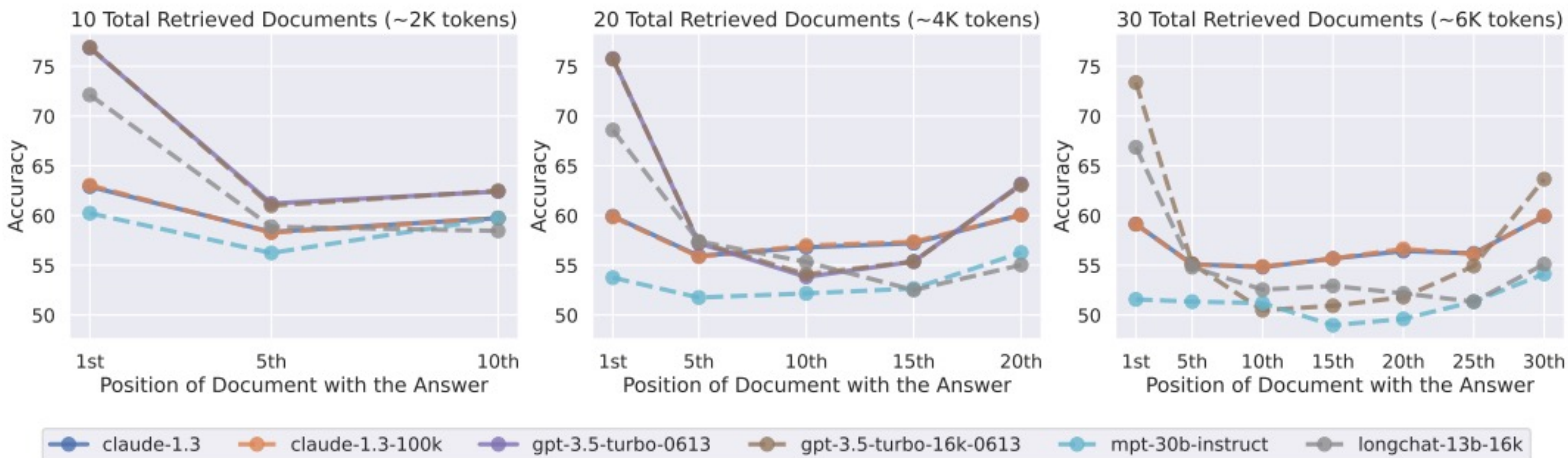
Models perform best when beginning or end of contexts!!!

Effect of position and context length on model performance



Known as **primacy bias** and **recency bias**

Effect of position and context length on model performance



Extended context models does not necessarily improve performances

We know models struggle to retrieve and use information in the middle of the input

Can they simply **retrieve** from input contexts?

Experiment setup

Input Context

```
Extract the value corresponding to the specified key in the JSON object below.

JSON data:
{"2a8d601d-1d69-4e64-9f90-8ad825a74195": "bb3ba2a5-7de8-434b-a86e-a88bb9fa7289",
 "a54e2eed-e625-4570-9f74-3624e77d6684": "d1ff29be-4e2a-4208-a182-0cea716be3d4",
 "9f4a92b9-5f69-4725-ba1e-403f08dea695": "703a7ce5-f17f-4e6d-b895-5836ba5ec71c",
 "52a9c80c-da51-4fc9-bf70-4a4901bc2ac3": "b2f8ea3d-4b1b-49e0-a141-b9823991ebeb",
 "f4eb1c53-af0a-4dc4-a3a5-c2d50851a178": "d733b0d2-6af3-44e1-8592-e5637fdb76fb"}

Key: "9f4a92b9-5f69-4725-ba1e-403f08dea695"
Corresponding value:
```

Desired Output

```
703a7ce5-f17f-4e6d-b895-5836ba5ec71c
```

Experiment setup

Input Context

```
Extract the value corresponding to the specified key in the JSON object below.

JSON data:
{"2a8d601d-1d69-4e64-9f90-8ad825a74195": "bb3ba2a5-7de8-434b-a86e-a88bb9fa7289",
 "a54e2eed-e625-4570-9f74-3624e77d6684": "d1ff29be-4e2a-4208-a182-0cea716be3d4",
 "9f4a92b9-5f69-4725-ba1e-403f08dea695": "703a7ce5-f17f-4e6d-b895-5836ba5ec71c",
 "52a9c80c-da51-4fc9-bf70-4a4901bc2ac3": "b2f8ea3d-4b1b-49e0-a141-b9823991ebeb",
 "f4eb1c53-af0a-4dc4-a3a5-c2d50851a178": "d733b0d2-6af3-44e1-8592-e5637fdb76fb"}

Key: "9f4a92b9-5f69-4725-ba1e-403f08dea695"
Corresponding value:
```

Desired Output

```
703a7ce5-f17f-4e6d-b895-5836ba5ec71c
```

Position #1

Experiment setup

Input Context

```
Extract the value corresponding to the specified key in the JSON object below.

JSON data:
{"2a8d601d-1d69-4e64-9f90-8ad825a74195": "bb3ba2a5-7de8-434b-a86e-a88bb9fa7289",
 "a54e2eed-e625-4570-9f74-3624e77d6684": "d1ff29be-4e2a-4208-a182-0cea716be3d4",
 "9f4a92b9-5f69-4725-ba1e-403f08dea695": "703a7ce5-f17f-4e6d-b895-5836ba5ec71c",
 "52a9c80c-da51-4fc9-bf70-4a4901bc2ac3": "b2f8ea3d-4b1b-49e0-a141-b9823991ebeb",
 "f4eb1c53-af0a-4dc4-a3a5-c2d50851a178": "d733b0d2-6af3-44e1-8592-e5637fdb76fb"}

Key: "9f4a92b9-5f69-4725-ba1e-403f08dea695"
Corresponding value:
```

Desired Output

```
703a7ce5-f17f-4e6d-b895-5836ba5ec71c
```

Experiment setup

Input Context

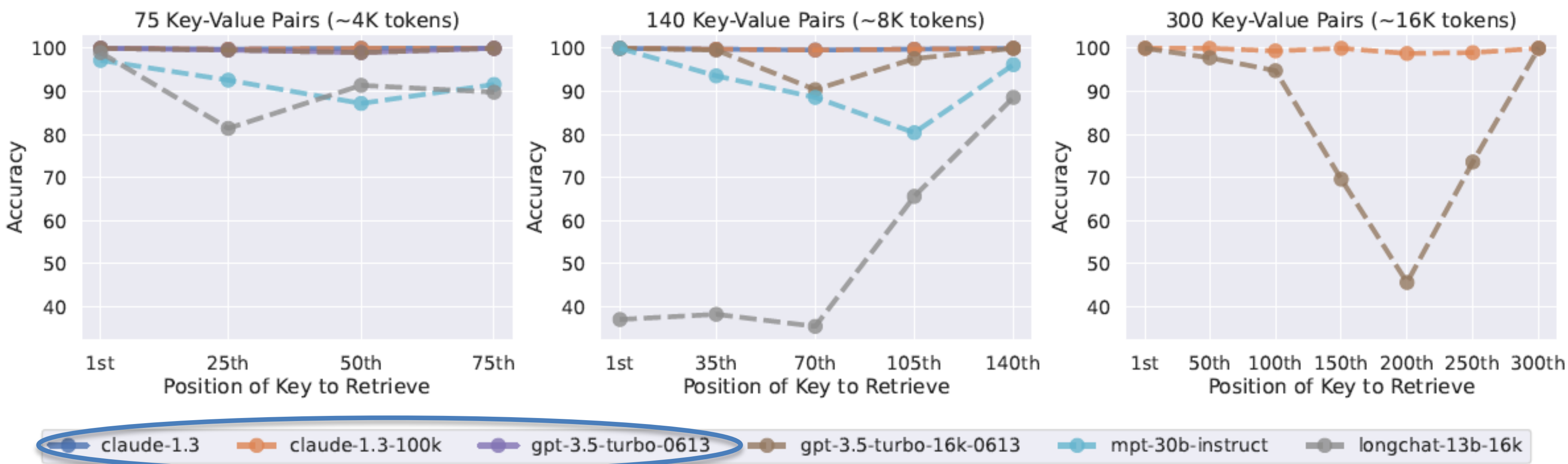
```
Extract the value corresponding to the specified key in the JSON object below.

JSON data:
{"2a8d601d-1d69-4e64-9f90-8ad825a74195": "bb3ba2a5-7de8-434b-a86e-a88bb9fa7289",
 "a54e2eed-e625-4570-9f74-3624e77d6684": "d1ff29be-4e2a-4208-a182-0cea716be3d4",
 "9f4a92b9-5f69-4725-ba1e-403f08dea695": "703a7ce5-f17f-4e6d-b895-5836ba5ec71c",
 "52a9c80c-da51-4fc9-bf70-4a4901bc2ac3": "b2f8ea3d-4b1b-49e0-a141-b9823991ebeb",
 "f4eb1c53-af0a-4dc4-a3a5-c2d50851a178": "d733b0d2-6af3-44e1-8592-e5637fdb76fb"}

Key: "9f4a92b9-5f69-4725-ba1e-403f08dea695"
Corresponding value:
```

Desired Output

```
703a7ce5-f17f-4e6d-b895-5836ba5ec71c
```

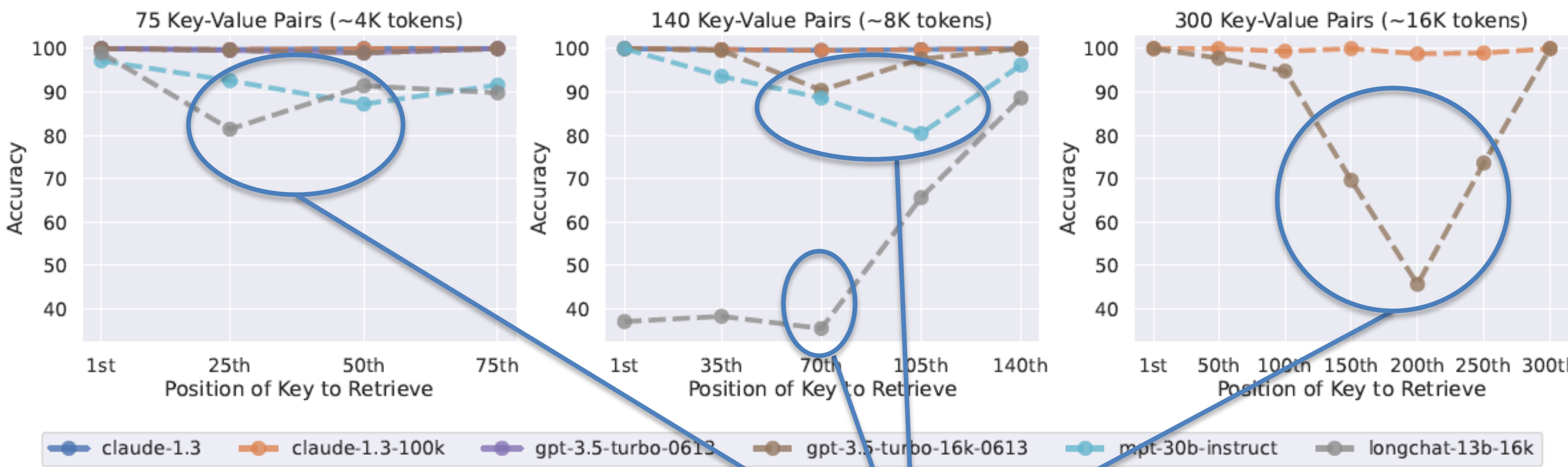
Tested with different lengths

Results



These models can achieve really good performances!!!

Results



Again, performance degrades in the middle of the input!!!

Why are models not robust to changes in the position of relevant information?

Effect of the model architecture?

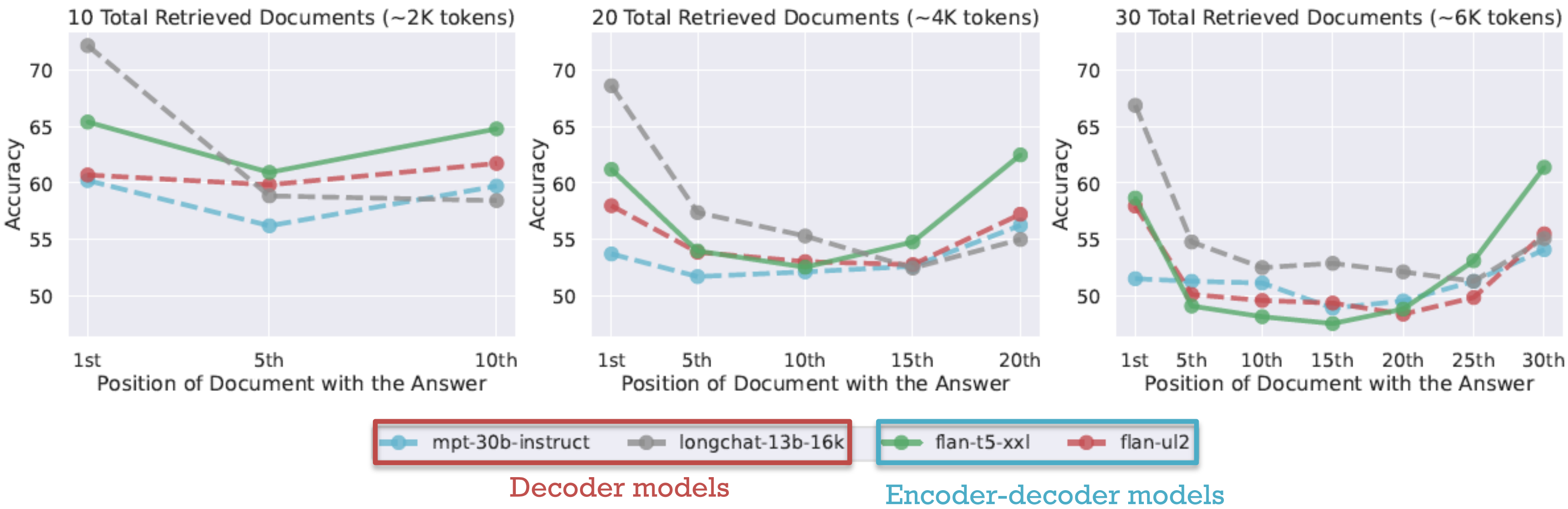
Effect of Query-aware contextualization

Effect of instruction fine-tuning

How do the models compare?

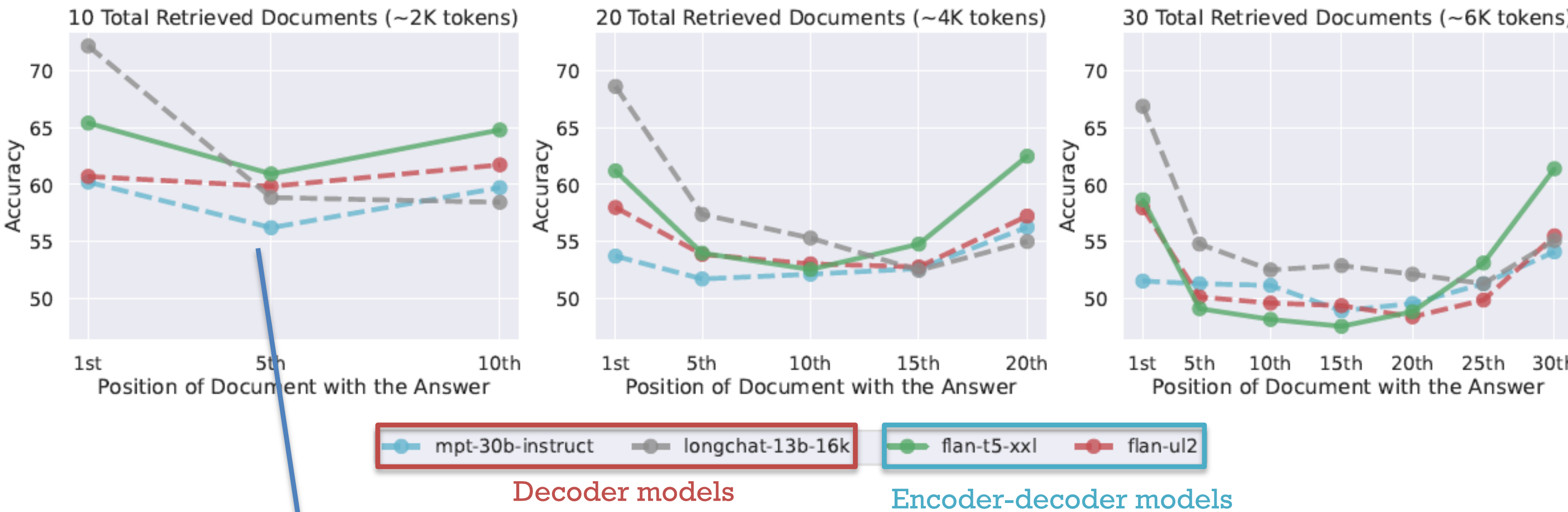
- Decoder model
  - longchat
  - MPT
- Encoder - decoder model
  - T5
  - UL2

Results



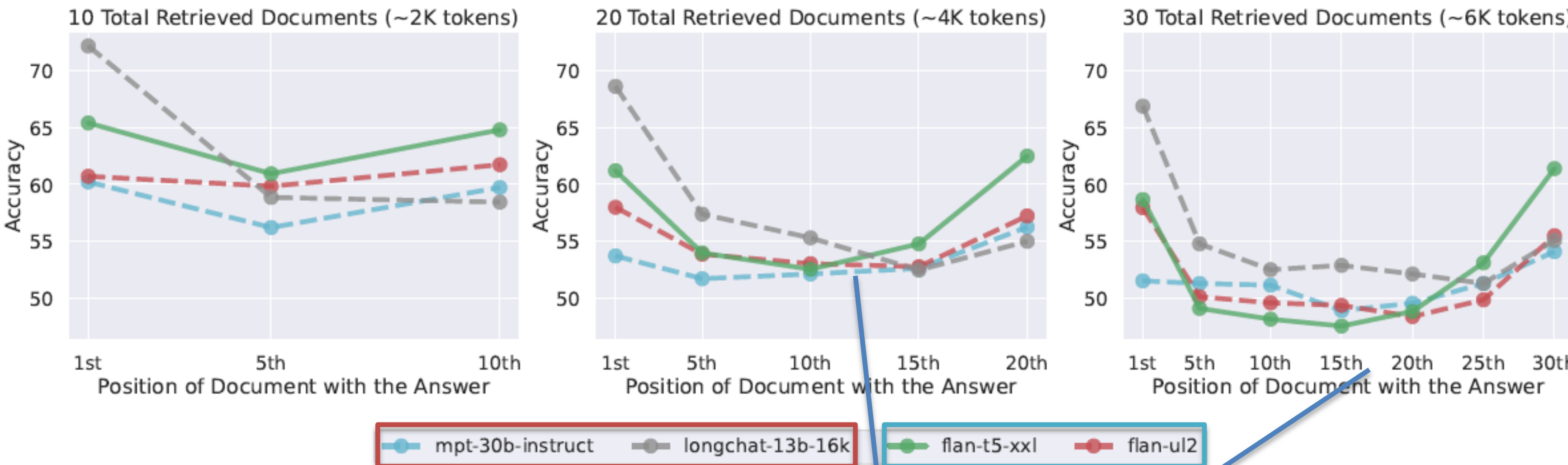
10 Total Retrieved Documents (~2K tokens)
20 Total Retrieved Documents (~4K tokens)
30 Total Retrieved Documents (~6K tokens)
Accuracy
70
65
60
55
50
1st
5th
10th
15th
20th
25th
30th
Position of Document with the Answer
mpt-30b-instruct
longchat-13b-16k
flan-t5-xxl
flan-ul2
Decoder models
Encoder-decoder models

Results



When shorter than their max sequence length
→ Robust to changes in positions

Results



Decoder models

Encoder-decoder models

Encoder-decoder models becomes more ‘U’ shaped for longer sequences

Effect of Query-aware contextualization?

Without Query-aware contextualization

Key-Value Pairs:
- "Germany": "Berlin"
- "France": "Paris"
- "Spain": "Madrid"

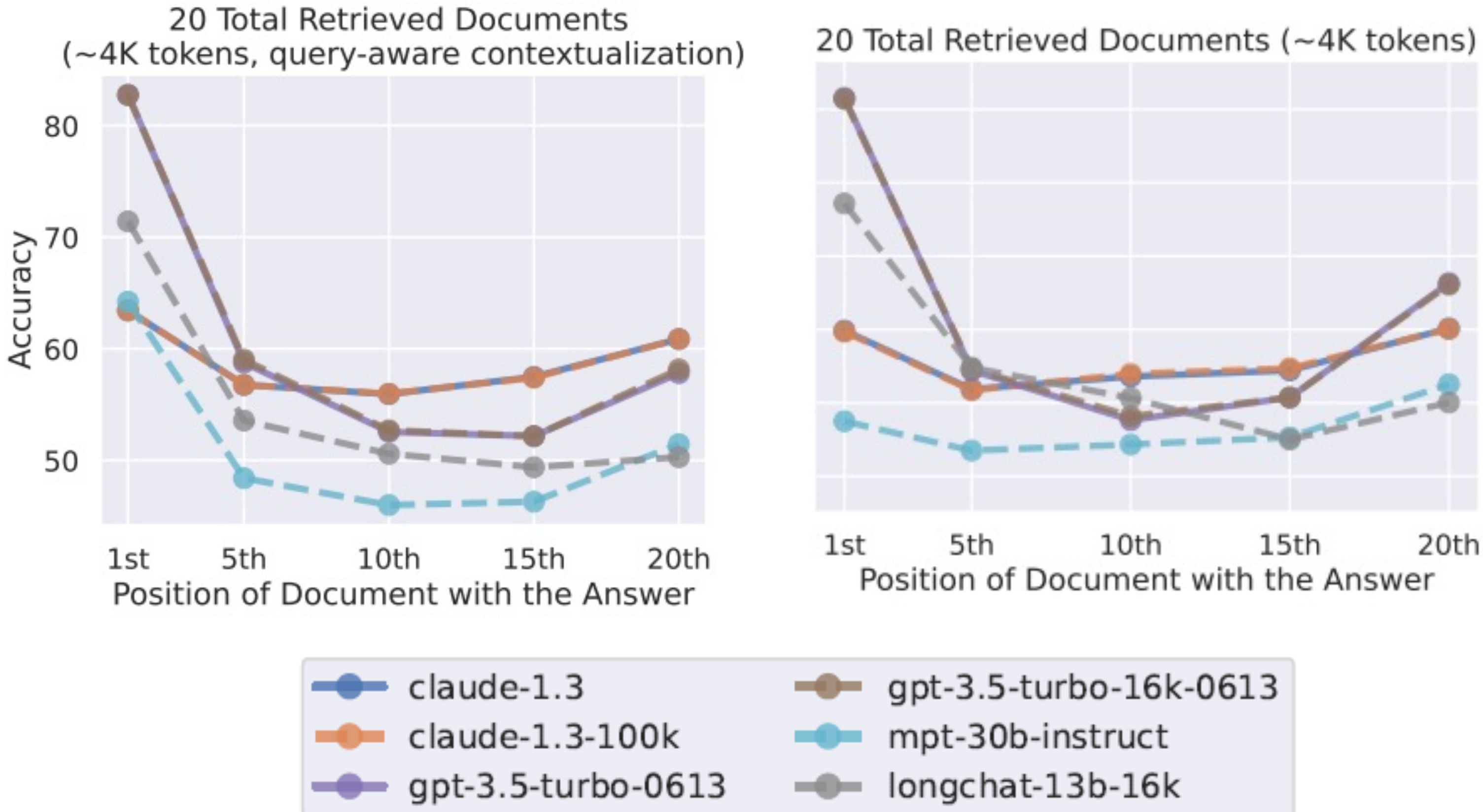
Query (Question): What is the capital of France?

With Query-aware contextualization

Query (Question): What is the capital of France?

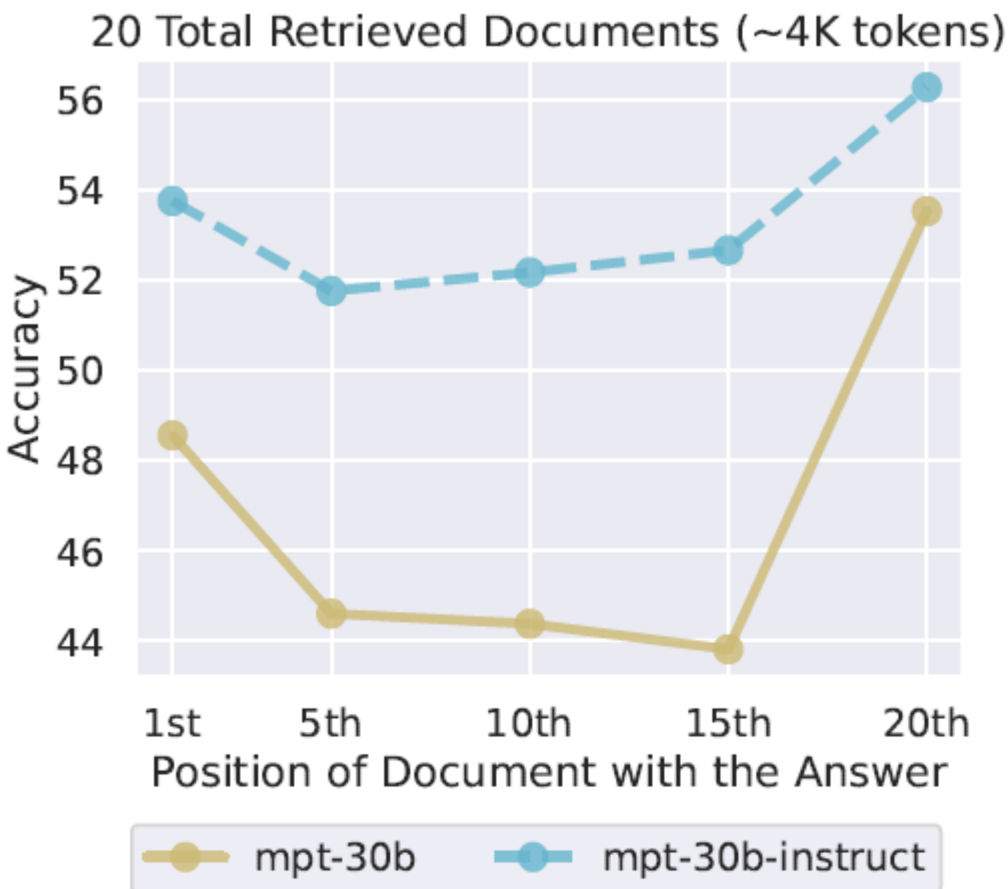
Key-Value Pairs:
- "Germany": "Berlin"
- "France": "Paris"
- "Spain": "Madrid"

Query-aware contextualization does NOT affect performance

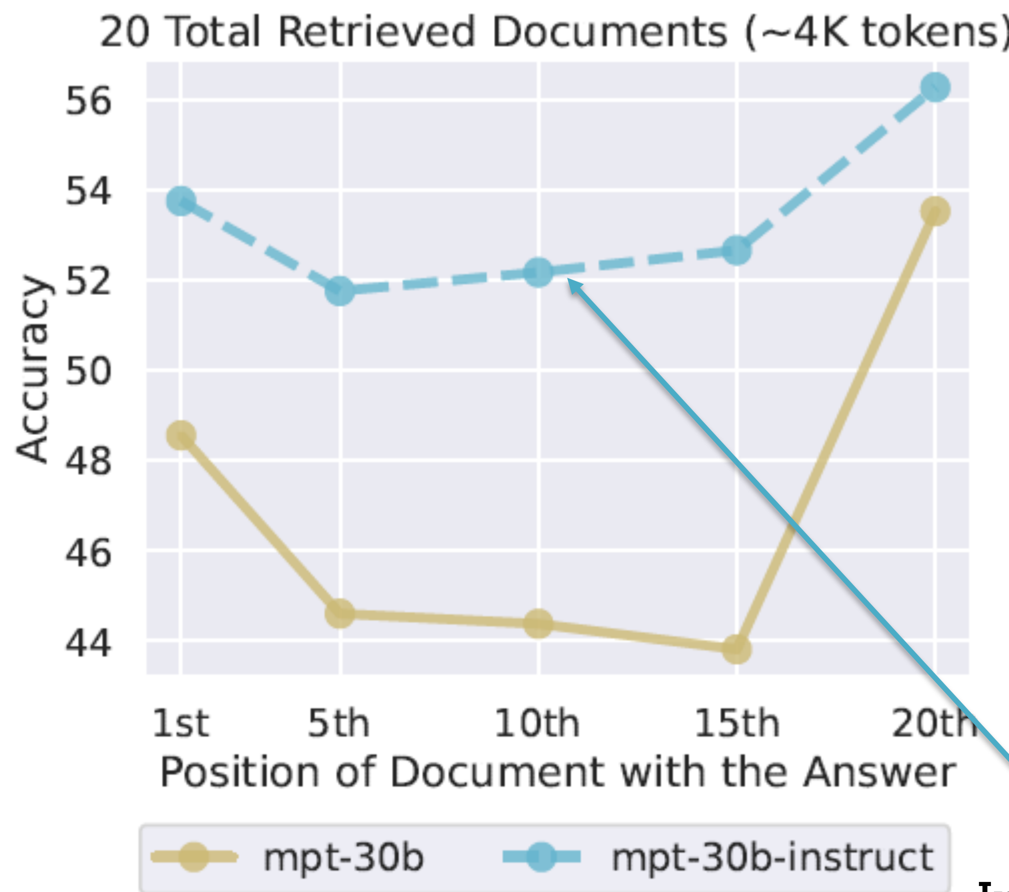


Not much difference!!!!

Effect of instruction fine-tuning???

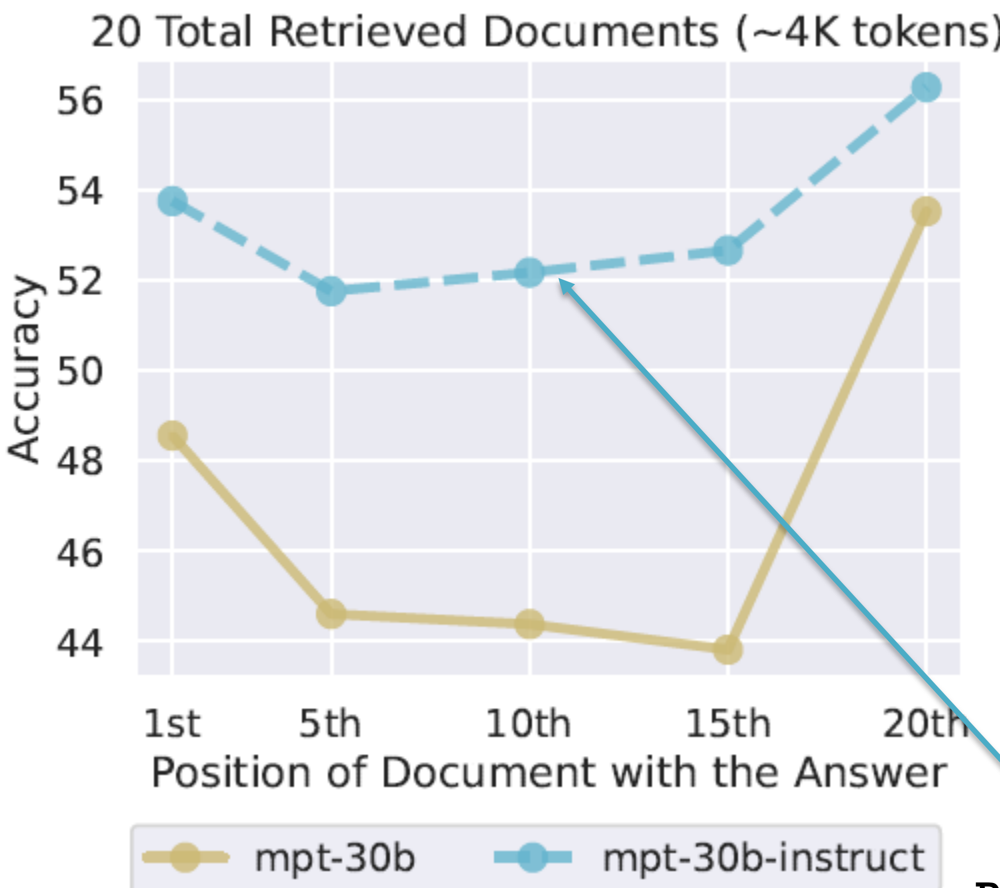


Effect of instruction fine-tuning???



Improvements from instruction tuning

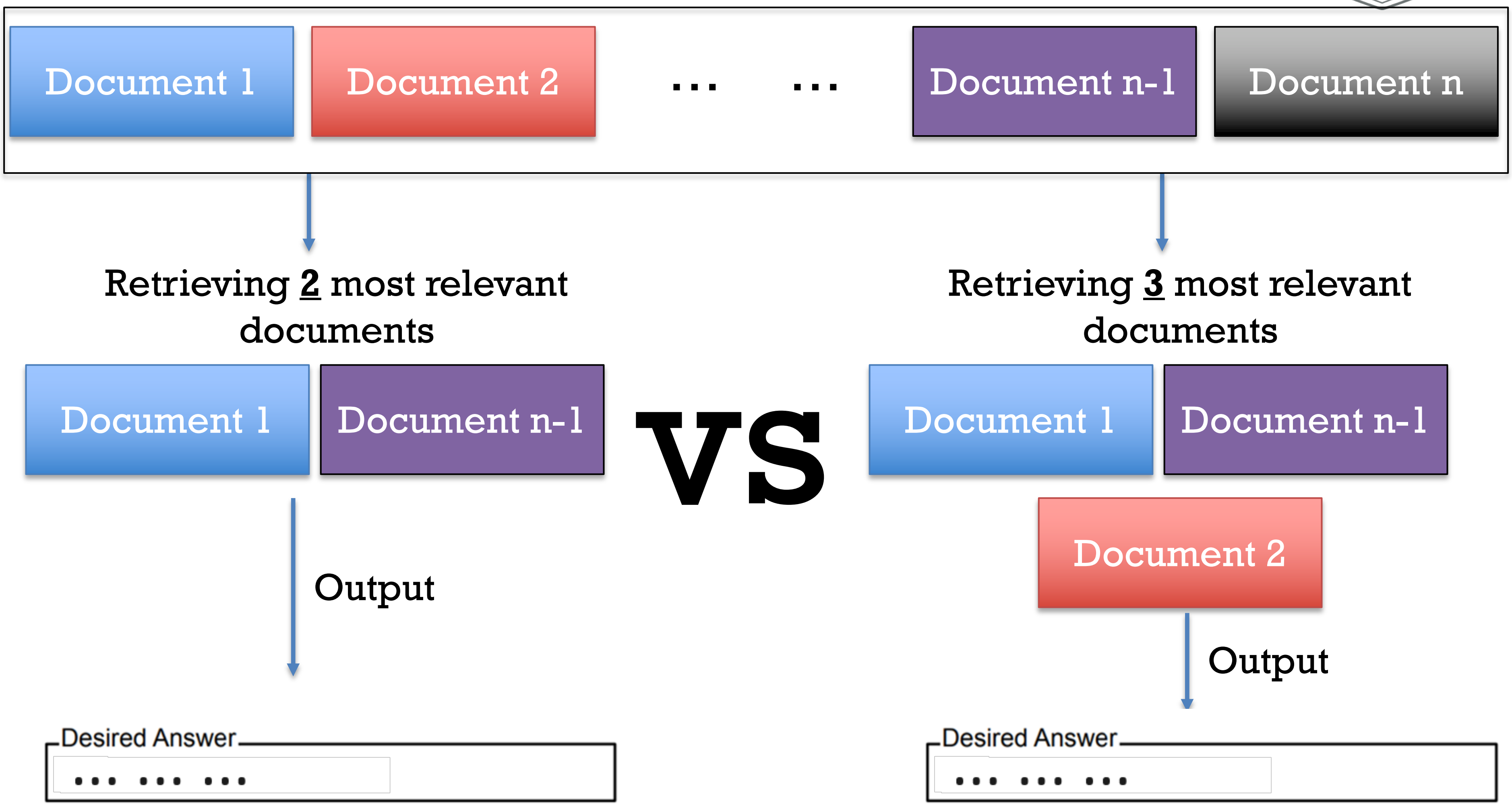
Effect of instruction fine-tuning???



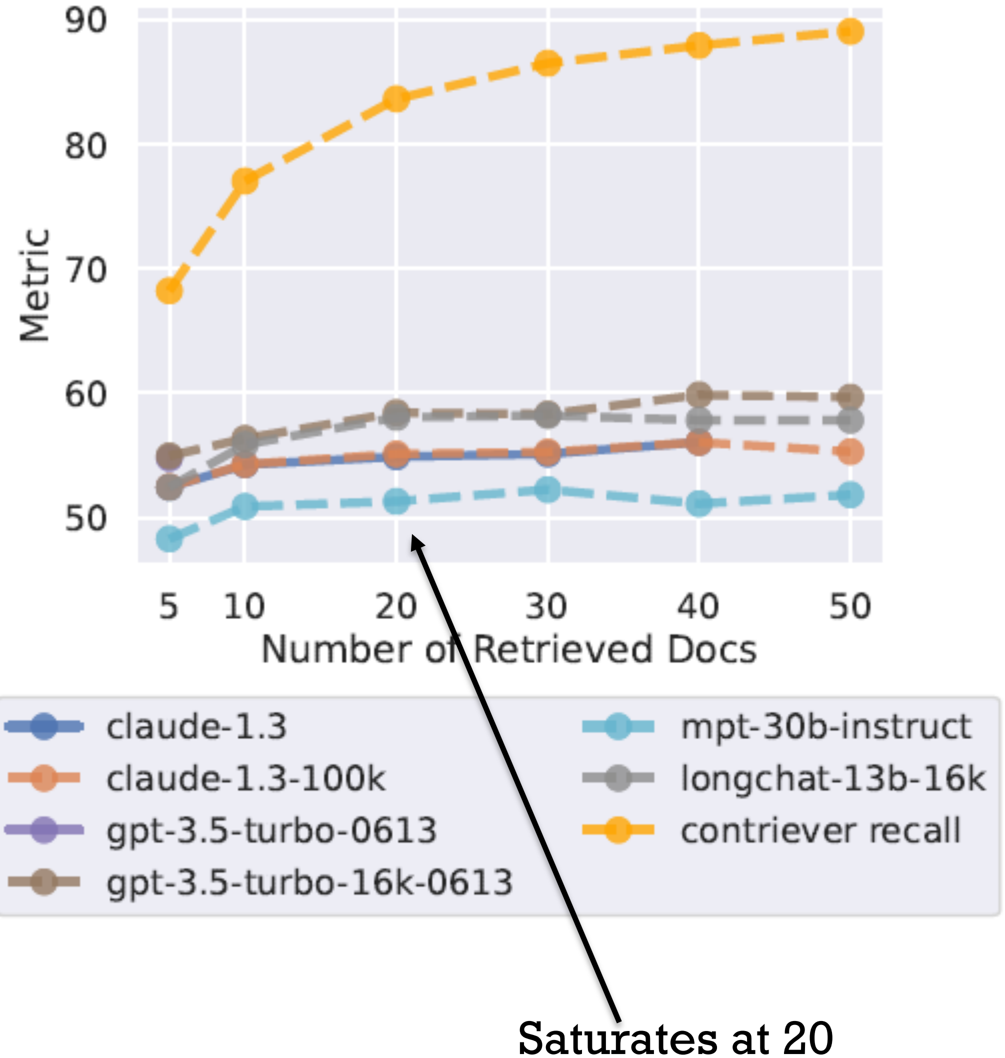
But ‘U’ shape is still present

Is more context better?

Experiment (pseudo-example)

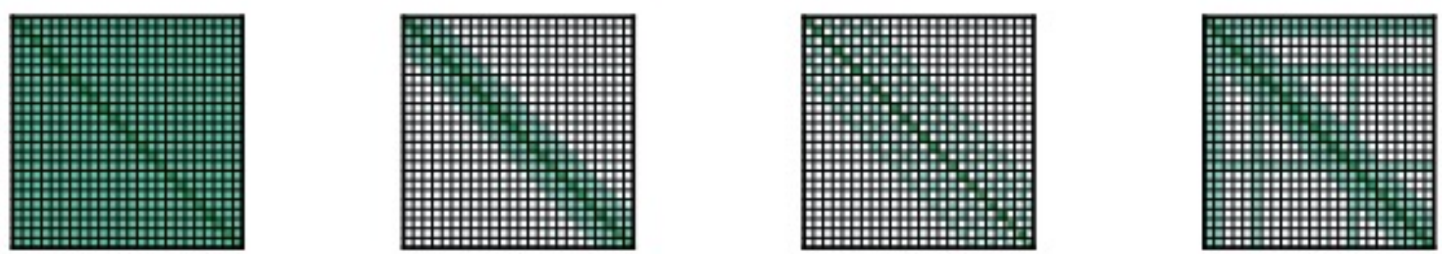


Number of retrieved documents on model performance



Summary

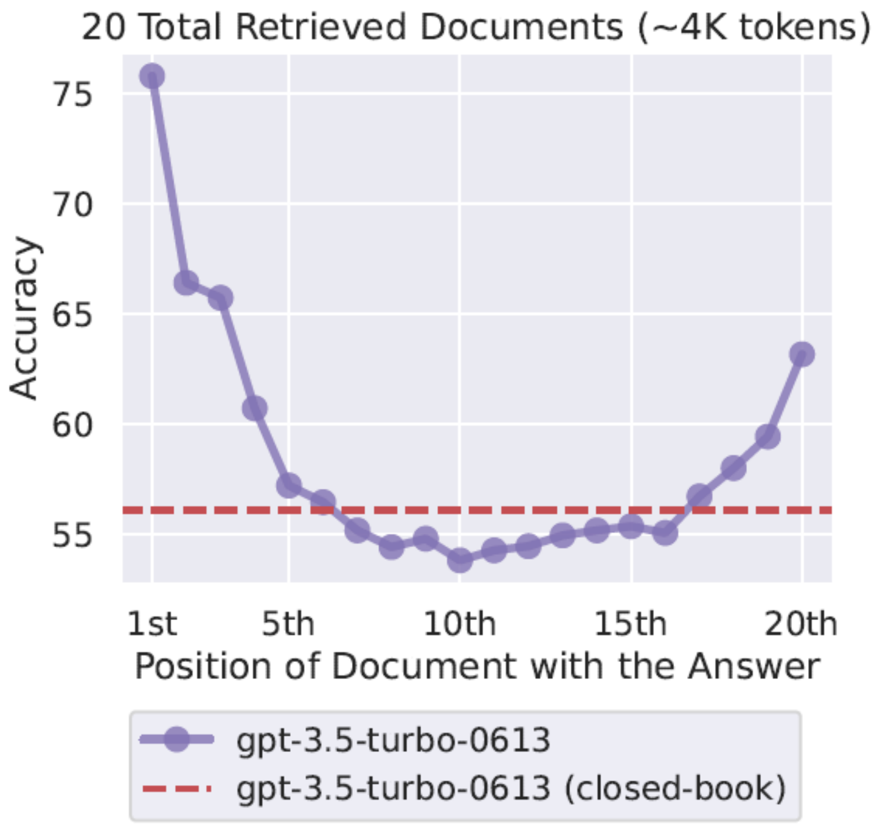
Demonstrated how LLMs tries to understand long context efficiently via Longformers



(a) Full n^2 attention (b) Sliding window attention (c) Dilated sliding window (d) Global+sliding window

Summary

- Demonstrated downsides and precautions to using long-contexts with LLMs.



Q&A

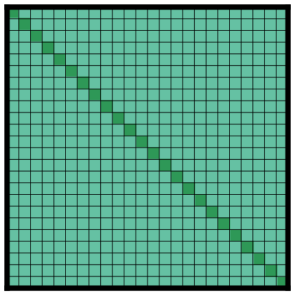
Question:

limitations to Longformer in capturing long-range dependencies compared to traditional full self-attention mechanisms?

Answer

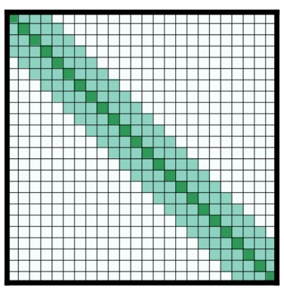
- It cannot capture long-range dependencies if the window size is too small!
- But large window size still requires more GPUs, even if it scales linearly!

Lets do a quick comparison



(a) Full n^2 attention

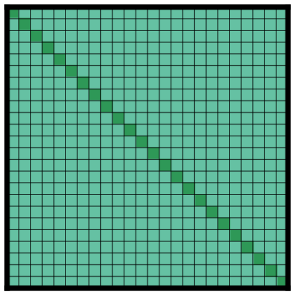
I __love__ Washington University. It is a great school!



(b) Sliding window attention

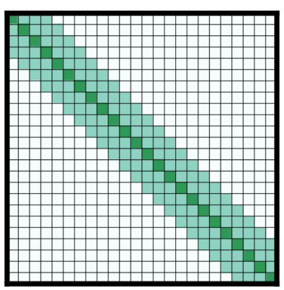
I __love__ Washington University. It is a great school!

Lets do a quick comparison



(a) Full n^2 attention

I __love__ Washington University. It is a great school!

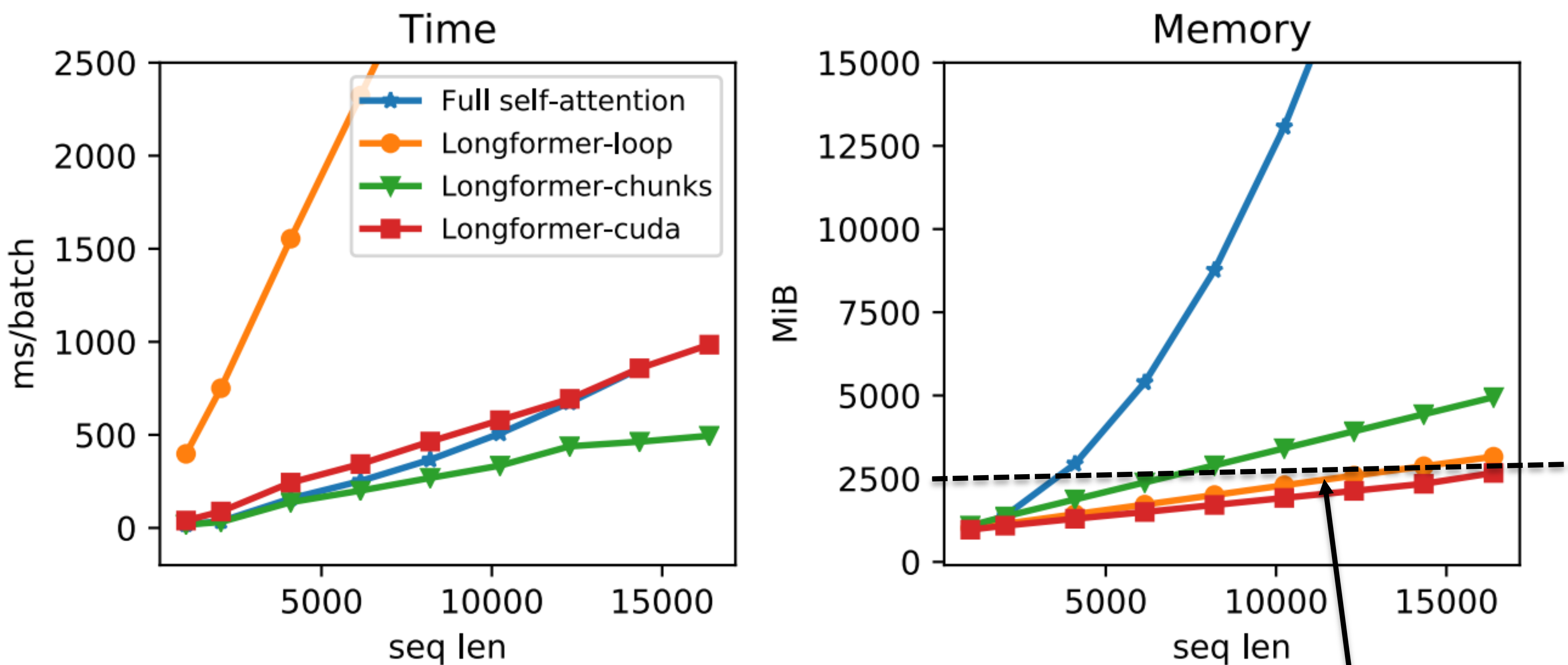


(b) Sliding window attention

I __love__ Washington University. It is a great school!

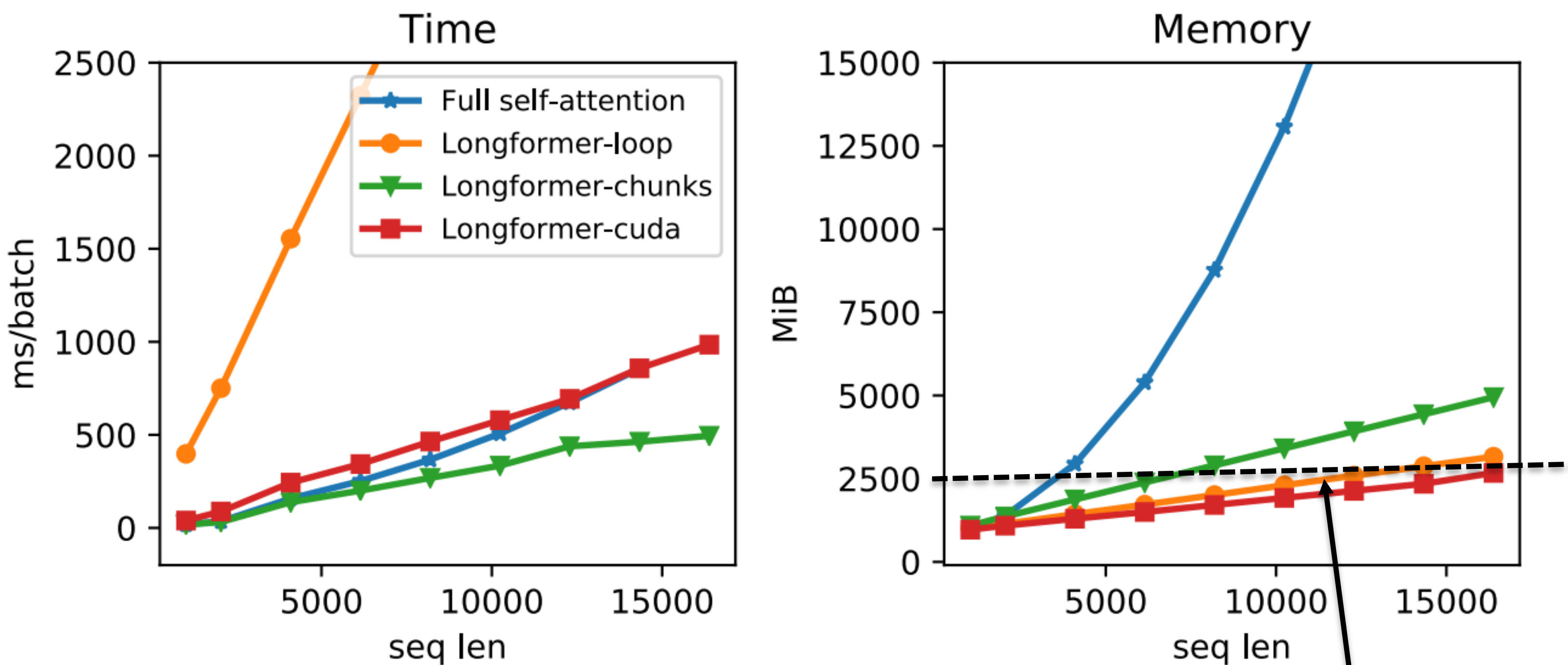
This will do a poorer job as it attends to a very small window size

Thinking practically: GPUs are expensive



Let us assume this is your GPU threshold

Thinking practically: GPUs are expensive



You can have a larger sequence length, but you will be bounded here

Question:

How do we mitigate this?

Rule of thumb

Only pre-trained longformers only when you have long texts (> 512 tokens)

- Don't use for the sake of it!

Maximize window size $w \approx$ max token length

- Pointless if your window size is small

Question:

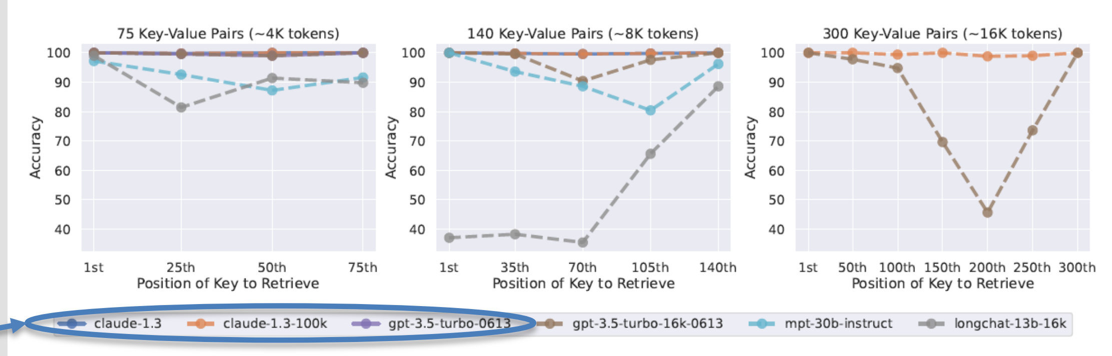
How can LongFormer be used to exploit the format of a document. For example how it can be used to process long json files, i.e. key-value pairs?

Question:

Is there some efficient way to exploit the format of a document and reduce computational cost?

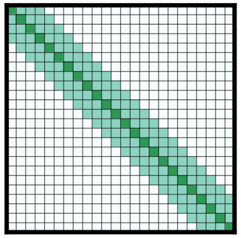
Answer

You may want to use these models instead!!!



But if you want to use Longformers, you may want to use the Autoregressive version of Longformers

Uses sliding window attention

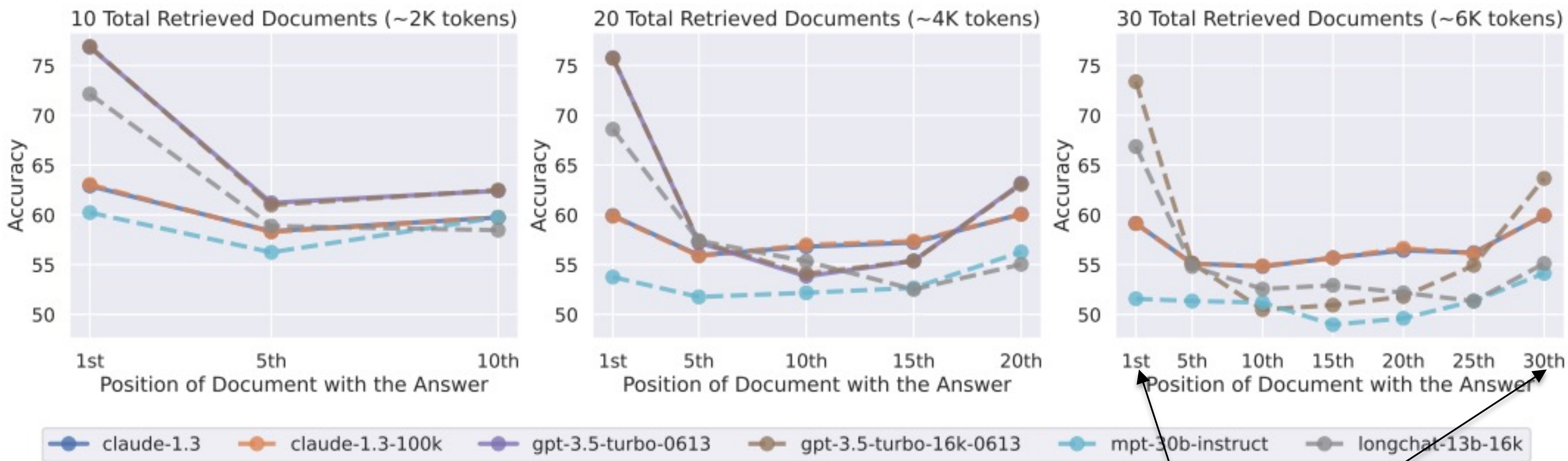


(b) Sliding window attention

Question:

How do current language models perform in tasks that require accessing and utilizing information from long input contexts, and what are the implications for the design of future long-context language models?

Effect of position and context length on model performance



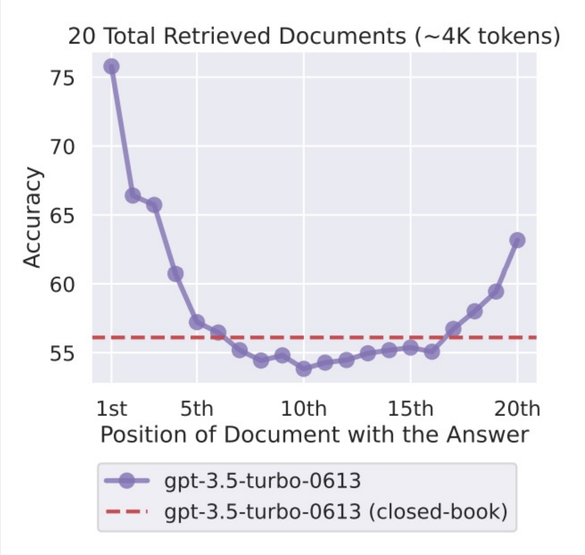
Models perform best when beginning or end of contexts!!!

Question:

For future language models tasked with long, constant interaction, is there a need to implement a temporary weight vector to enhance performance and context utilization?

Answer:

Open question, but the 2^{nd} paper provides us with a framework in potentially doing so.



Thank you!

Supplemental slides

Global attention for longformer

Linear Projections for Global Attention Recall that given the linear projections Q, K, V, the Transformer model (Vaswani et al., 2017) computes attention scores as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (1)$$

We use two sets of projections, Q_s, K_s, V_s to compute attention scores of sliding window attention, and Q_g, K_g, V_g to compute attention scores for the global attention. The additional projections provide flexibility to model the different types of attention, which we show is critical for best performance on downstream tasks. Q_g, K_g, V_g are all initialized with values that match Q_s, K_s, V_s.

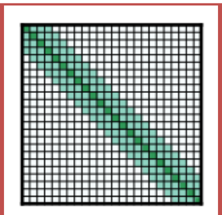
Pretraining and fine-tuning?

Pretraining

Can take process sequences of **4096** tokens long

Continued pre-training from **RoBERTa** model

Sliding window of **w=512**



Positional embeddings initialized from RoBERTa

- **To preserve local structure**

Model	base	large
RoBERTa (seqlen: 512)	1.846	1.496
Longformer (seqlen: 4,096)	10.299	8.738
+ copy position embeddings	1.957	1.597
+ 2K gradient updates	1.753	1.414
+ 65K gradient updates	1.705	1.358
Longformer (train extra pos. embed. only)	1.850	1.504

↓ is better

Table 5: MLM BPC for RoBERTa and various pretrained Longformer configurations.

Finetuning on 3 tasks:

Question Answering*

Coreference resolution

Document classification*

* Global attention is used on these tasks

Achieves amazing performances on multiple tasks

	QA			Coref.	Classification	
Model	WikiHop	TriviaQA	HotpotQA	OntoNotes	IMDB	Hyperpartisan
RoBERTa-base	72.4	74.3	63.5	78.4	95.3	87.4
Longformer-base	**75.0**	**75.2**	**64.4**	**78.6**	**95.7**	**94.8**

Table 7: Summary of finetuning results on QA, coreference resolution, and document classification. Results are on the development sets comparing our Longformer-base with RoBERTa-base. TriviaQA, Hyperpartisan metrics are F1, WikiHop and IMDB use accuracy, HotpotQA is joint F1, OntoNotes is average F1.

Achieved state-of-the-art performance for Q&A (at that time…)

Model	WikiHop	TriviaQA	HotpotQA
Current* SOTA	78.3	73.3	**74.2**
Longformer-large	**81.9**	**77.3**	73.2

Table 8: Leaderboard results of Longformer-large at time of submission (May 2020). All numbers are F1 scores.

Achieved competitive performance (at that time)

Model	ans.	supp.	joint
TAP 2 (ensemble) (Glaß et al., 2019)	79.8	86.7	70.7
SAE (Tu et al., 2019)	79.6	86.7	71.4
Quark (dev) (Groeneveld et al., 2020)	81.2	87.0	72.3
C2F Reader (Shao et al., 2020)	81.2	87.6	72.8
Longformer-large	81.3	88.3	73.2
ETC-large† (Ainslie et al., 2020)	81.2	89.1	73.6
GSAN-large†	81.6	88.7	73.9
HGN-large (Fang et al., 2020)	82.2	88.5	74.2

Table 9: HotpotQA results in distractor setting test set. Quark's test results are not available. All numbers are F1 scores. † shows contemporaneous leaderboard submissions.

Note: GNN-based models

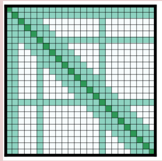
Ablation studies

Model	Accuracy / Δ
Longformer (seqlen: 4,096)	73.8
RoBERTa-base (seqlen: 512)	72.4 / -1.4
Longformer (seqlen: 4,096, 15 epochs)	75.0 / +1.2
Longformer (seqlen: 512, attention: n^2)	71.7 / -2.1
Longformer (seqlen: 2,048)	73.1 / -0.7
Longformer (no MLM pretraining)	73.2 / -0.6
Longformer (no linear proj.)	72.2 / -1.6
Longformer (no linear proj. no global atten.)	65.5 / -8.3
Longformer (pretrain extra position embed. only)	73.5 / -0.3

Table 10: WikiHop development set ablations

Encoder-Decoder model?

Training



Uses local + global attention

Parameters initialized with the BART model

Can accommodate 16,000 tokens
(16× more than BART!!!)

Results

	R-1	R-2	R-L
Discourse-aware (2018)	35.80	11.05	31.80
Extr-Abst-TLM (2020)	41.62	14.69	38.03
Dancer (2020)	42.70	16.54	38.44
Pegasus (2020)	44.21	16.95	38.83
LED-large (seqlen: 4,096) (ours)	44.40	17.94	39.76
BigBird (seqlen: 4,096) (2020)	**46.63**	19.02	41.77
LED-large (seqlen: 16,384) (ours)	**46.63**	**19.62**	**41.83**

Table 11: Summarization results of Longformer-Encoder-Decoder (LED) on the arXiv dataset. Metrics from left to right are ROUGE-1, ROUGE-2 and ROUGE-L.

Works particularly well with longer input sizes!!!

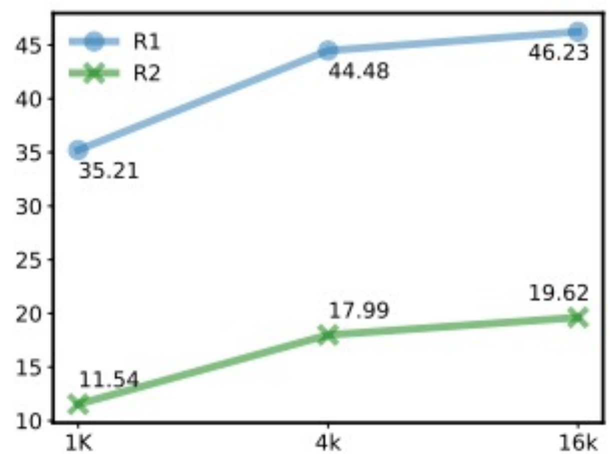


Figure 3: ROUGE-1 and ROUGE-2 of LED when varying the input size (arXiv validation set).

Closed book vs oracle

Model	Closed-Book	Oracle
LongChat-13B (16K)	35.0%	83.4%
MPT-30B-Instruct	31.5%	81.9%
GPT-3.5-Turbo	56.1%	88.3%
GPT-3.5-Turbo (16K)	56.0%	88.6%
Claude-1.3	48.3%	76.1%
Claude-1.3 (100K)	48.2%	76.4%

Table 1: Closed-book and oracle accuracy of language models on the multi-document question answering task.