Adapting the Transformer Attention Mechanism for Efficient and Effective Information Retrieval

Tübingen, 06.06.2025

Ferdinand Schlatt

ferdinand.schlatt@uni-jena.de

Friedrich-Schiller-Universität Jena

FRIEDRICH-SCHILLER-
UNIVERSITÄT
JENA

Adapting Attention

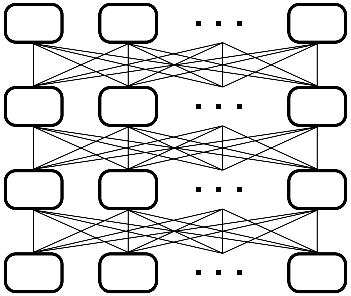
Standard Encoder Models for NLP

Transformer-based models are designed be as flexible as possible.

Adapting Attention

Standard Encoder Models for NLP

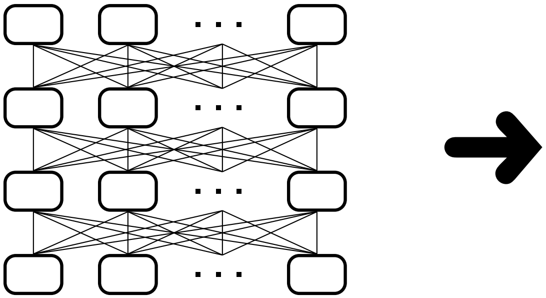
Transformer-based models are designed be as flexible as possible.



Adapting Attention

Standard Encoder Models for NLP

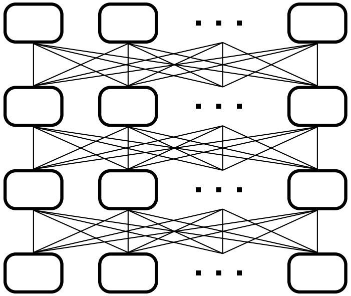
Transformer-based models are designed be as flexible as possible.

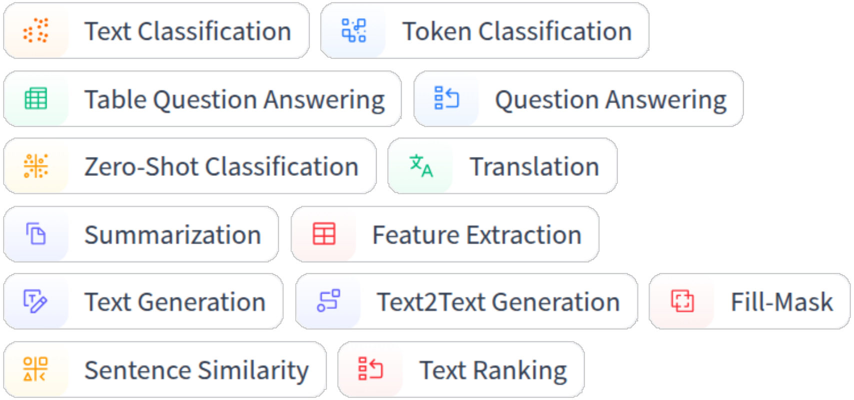


Adapting Attention

Standard Encoder Models for NLP

Transformer-based models are designed be as flexible as possible.





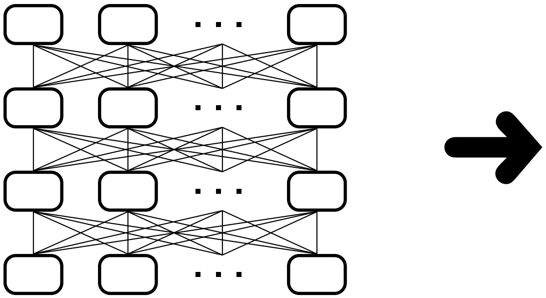
Text Classification
Token Classification
Table Question Answering
Question Answering
Zero-Shot Classification
Translation
Summarization
Feature Extraction
Text Generation
Text2Text Generation
Fill-Mask
Sentence Similarity
Text Ranking

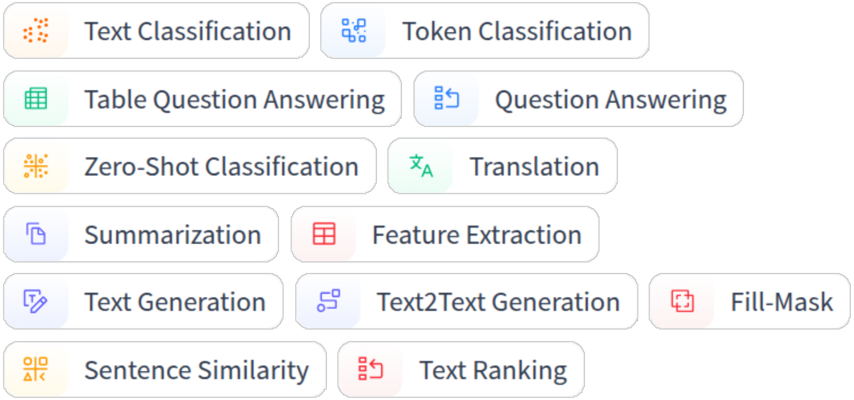
https://huggingface.co/models

Adapting Attention

Standard Encoder Models for NLP

Transformer-based models are designed be as flexible as possible.





Text Classification | Token Classification

Table Question Answering | Question Answering

Zero-Shot Classification | Translation

Summarization | Feature Extraction

Text Generation | Text2Text Generation | Fill-Mask

Sentence Similarity | Text Ranking

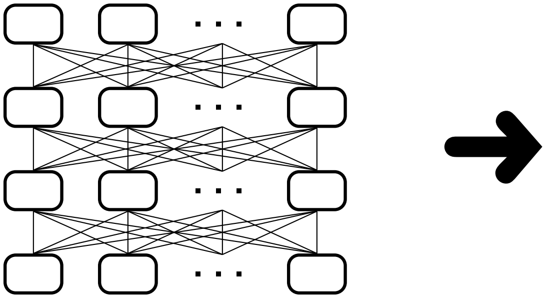
`https://huggingface.co/models`

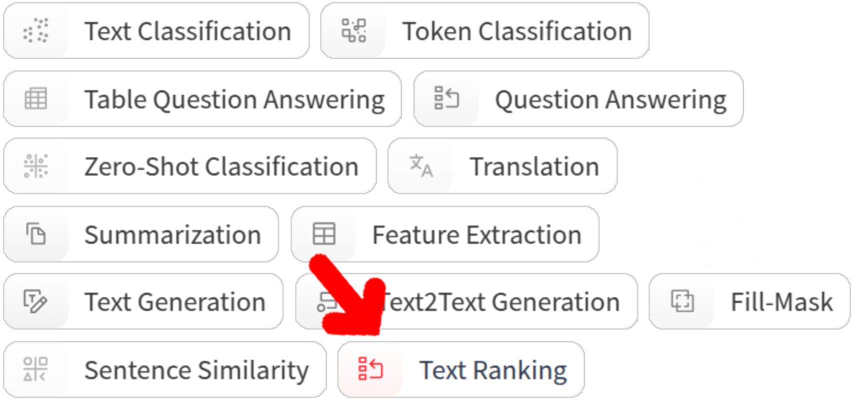
Can we improve performance by “fine-tuning” the attention mechanism?

Adapting Attention

Standard Encoder Models for NLP

Transformer-based models are designed be as flexible as possible.





Text Classification

Token Classification

Table Question Answering

Question Answering

Zero-Shot Classification

Translation

Summarization

Feature Extraction

Text Generation

Text2Text Generation

Fill-Mask

Sentence Similarity

Text Ranking

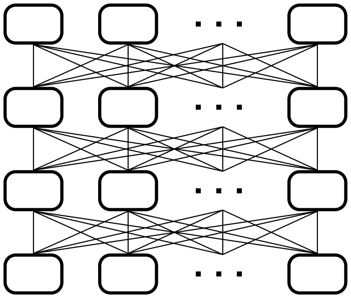
https://huggingface.co/models

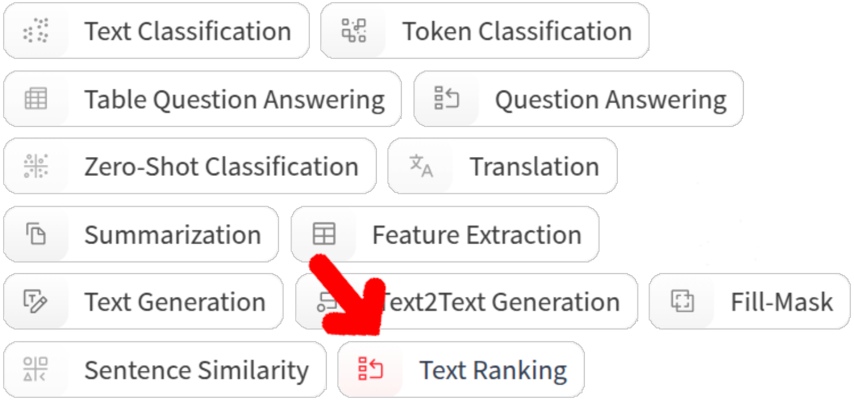
Can we improve performance by “fine-tuning” the attention mechanism?

Adapting Attention

Standard Encoder Models for NLP

Transformer-based models are designed be as flexible as possible.





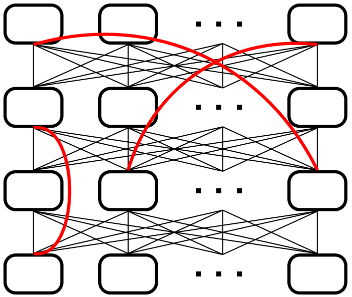
https://huggingface.co/models

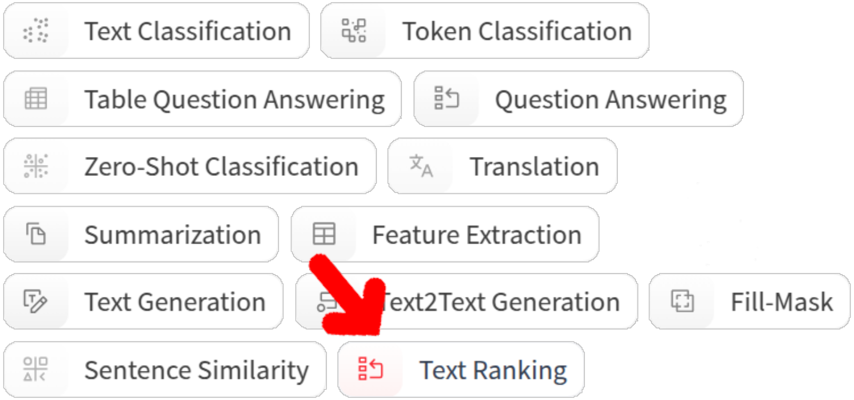
Can we improve performance by “fine-tuning” the attention mechanism?

Adapting Attention

Standard Encoder Models for NLP

Transformer-based models are designed be as flexible as possible.





https://huggingface.co/models

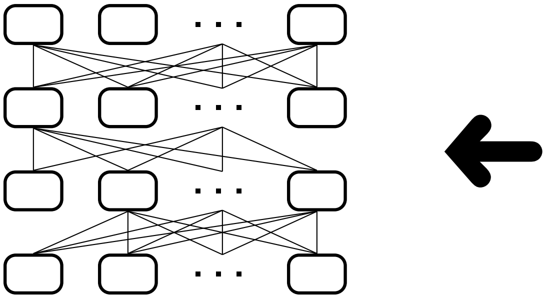
Can we improve performance by “fine-tuning” the attention mechanism?

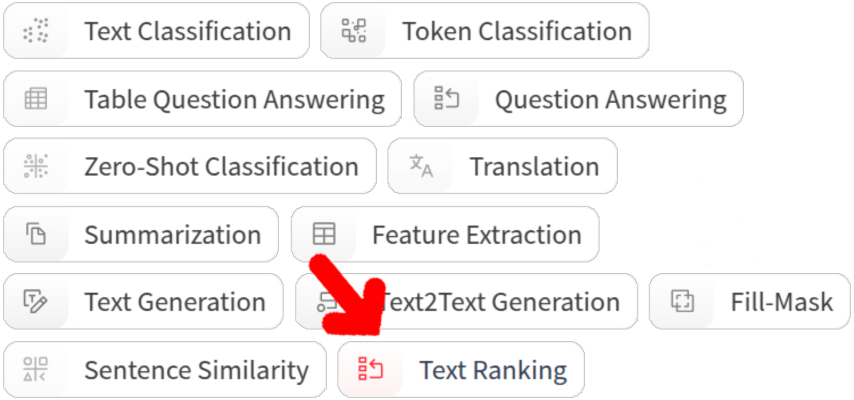
- We could add attention to make the model more effective . . .

Adapting Attention

Standard Encoder Models for NLP

Transformer-based models are designed be as flexible as possible.





Text Classification
Token Classification
Table Question Answering
Question Answering
Zero-Shot Classification
Translation
Summarization
Feature Extraction
Text Generation
Text2Text Generation
Fill-Mask
Sentence Similarity
Text Ranking

https://huggingface.co/models

Can we improve performance by “fine-tuning” the attention mechanism?

- We could add attention to make the model more effective . . .
- . . . or remove attention to make the model more efficient

Set-Encoder

Comparing Pointwise, Pairwise, and Listwise Cross-Encoders

Query 🔍 learn python

Documents 🗎 Python is a great language to learn.
Pythons live in the rainforest.
Guido van Rossum invented Python.

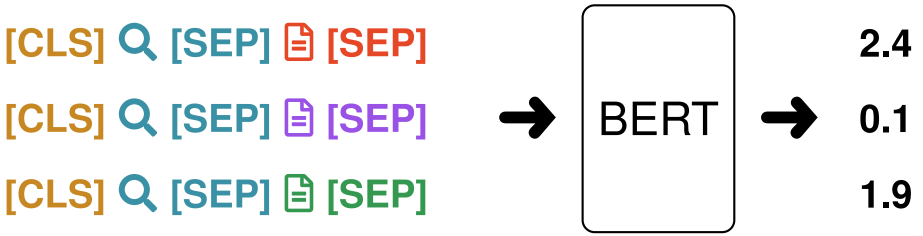
Set-Encoder

Comparing Pointwise, Pairwise, and Listwise Cross-Encoders

Query 🔍 learn python

Documents 📄 Python is a great language to learn.
Pythons live in the rainforest.
Guido van Rossum invented Python.

monoBERT (pointwise) [Nogueira and Cho, arXiv’19]



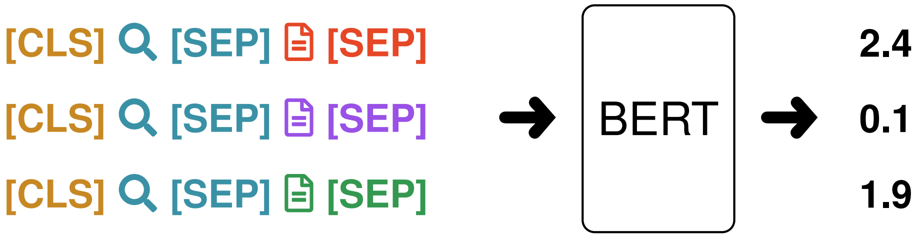
Set-Encoder

Comparing Pointwise, Pairwise, and Listwise Cross-Encoders

Query 🔍 learn python

Documents 📄 Python is a great language to learn.
Pythons live in the rainforest.
Guido van Rossum invented Python.

monoBERT (pointwise) [Nogueira and Cho, arXiv'19]


[CLS] 🔍 [SEP] 📄 [SEP]
[CLS] 🔍 [SEP] 📄 [SEP]
[CLS] 🔍 [SEP] 📄 [SEP]

Issue: The model scores each document independently.

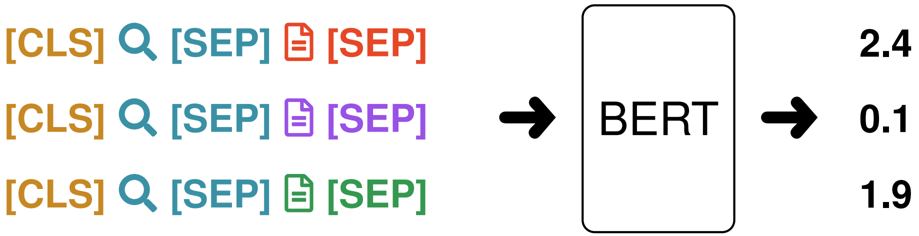
Set-Encoder

Comparing Pointwise, Pairwise, and Listwise Cross-Encoders

Query 🔍 learn python

Documents 🗎 Python is a great language to learn.
Pythons live in the rainforest.
Guido van Rossum invented Python.

monoBERT (pointwise) [Nogueira and Cho, arXiv'19]



[CLS] 🔍 [SEP] 🗎 [SEP]
[CLS] 🔍 [SEP] 🗎 [SEP]
[CLS] 🔍 [SEP] 🗎 [SEP]

→ BERT →

2.4
0.1
1.9

Issue: The model scores each document independently.

➜ Listwise (and pairwise) models enable interactions between documents.

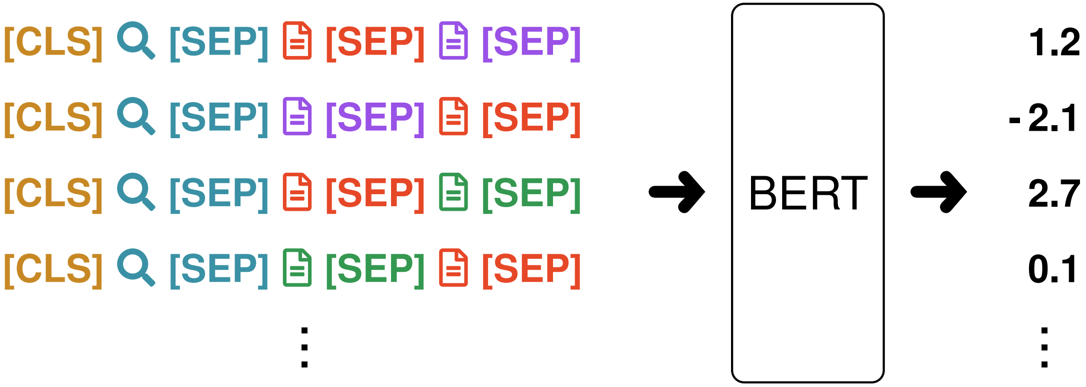
Set-Encoder

Comparing Pointwise, Pairwise, and Listwise Cross-Encoders

Query 🔍 learn python

Documents 📄 Python is a great language to learn.
Pythons live in the rainforest.
Guido van Rossum invented Python.

duoBERT (pairwise) [Nogueira et al., arXiv’20]



[CLS] 🔍 [SEP] 📄 [SEP] 📄 [SEP]
[CLS] 🔍 [SEP] 📄 [SEP] 📄 [SEP]
[CLS] 🔍 [SEP] 📄 [SEP] 📄 [SEP]
[CLS] 🔍 [SEP] 📄 [SEP] 📄 [SEP]
⋮

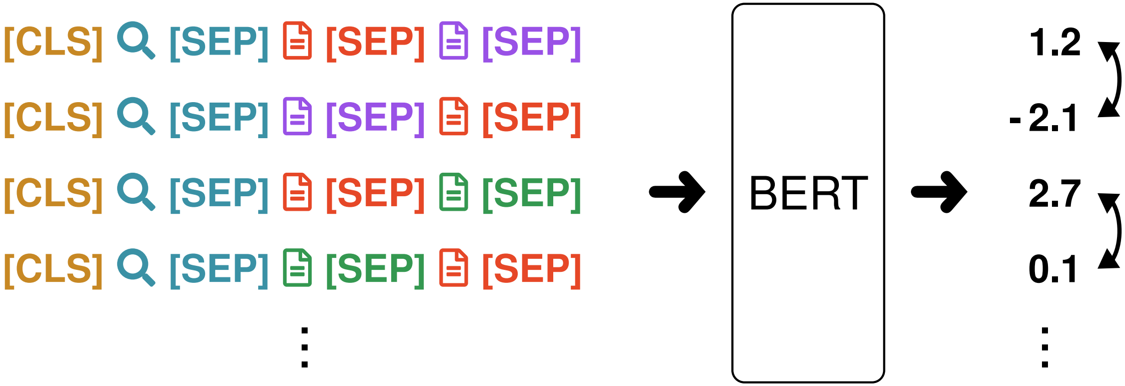
Set-Encoder

Comparing Pointwise, Pairwise, and Listwise Cross-Encoders

Query 🔍 learn python

Documents 📄 Python is a great language to learn.
Pythons live in the rainforest.
Guido van Rossum invented Python.

duoBERT (pairwise) [Nogueira et al., arXiv'20]



[CLS] 🔍 [SEP] 📄 [SEP] 📄 [SEP]
[CLS] 🔍 [SEP] 📄 [SEP] 📄 [SEP]
[CLS] 🔍 [SEP] 📄 [SEP] 📄 [SEP]
[CLS] 🔍 [SEP] 📄 [SEP] 📄 [SEP]
⋮

Issue: Relevance scores are not symmetric.

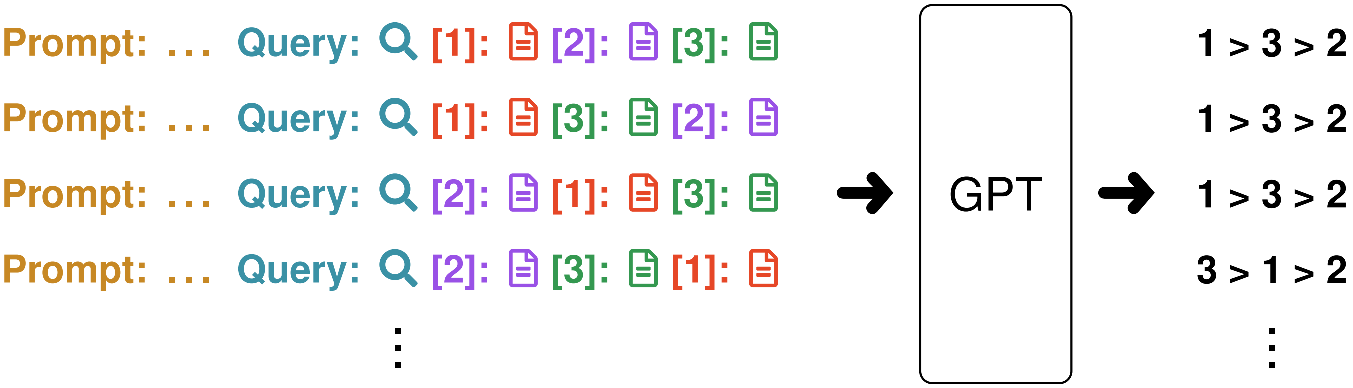
Set-Encoder

Comparing Pointwise, Pairwise, and Listwise Cross-Encoders

Query 🔍 learn python

Documents 🗎
Python is a great language to learn.
Pythons live in the rainforest.
Guido van Rossum invented Python.

RankGPT (listwise) [Sun et al., EMNLP’23]



Prompt: . . . Query: 🔍 [1]: 🗎 [2]: 🗎 [3]: 🗎
Prompt: . . . Query: 🔍 [1]: 🗎 [3]: 🗎 [2]: 🗎
Prompt: . . . Query: 🔍 [2]: 🗎 [1]: 🗎 [3]: 🗎
Prompt: . . . Query: 🔍 [2]: 🗎 [3]: 🗎 [1]: 🗎
⋮

→ GPT →

1 > 3 > 2
1 > 3 > 2
1 > 3 > 2
3 > 1 > 2
⋮

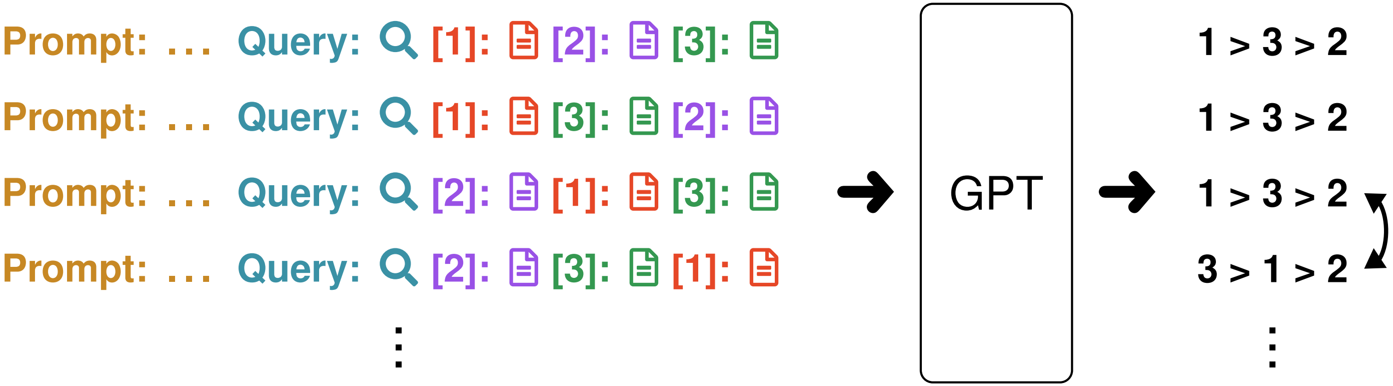
Set-Encoder

Comparing Pointwise, Pairwise, and Listwise Cross-Encoders

Query 🔍 learn python

Documents 📄 Python is a great language to learn.
Pythons live in the rainforest.
Guido van Rossum invented Python.

RankGPT (listwise) [Sun et al., EMNLP’23]



Prompt: . . . Query: 🔍 [1]: 📄 [2]: 📄 [3]: 📄

Prompt: . . . Query: 🔍 [1]: 📄 [3]: 📄 [2]: 📄

Prompt: . . . Query: 🔍 [2]: 📄 [1]: 📄 [3]: 📄

Prompt: . . . Query: 🔍 [2]: 📄 [3]: 📄 [1]: 📄

⋮

→ GPT →

1 > 3 > 2

1 > 3 > 2

1 > 3 > 2

3 > 1 > 2

⋮

Issue: Relevance preference order is not consistent.

Set-Encoder

Attention Mechanism

[CLS] learn python [SEP] Python is a great language to learn . [SEP]

[CLS] learn python [SEP] Pythons live in the rainforest . [SEP]

[CLS] learn python [SEP] Guido van Rossum invented Python . [SEP]

Set-Encoder

Attention Mechanism

[CLS] learn python [SEP] Python is a great language to learn . [SEP]

[CLS] learn python [SEP] Pythons live in the rainforest . [SEP]

[CLS] learn python [SEP] Guido van Rossum invented Python . [SEP]

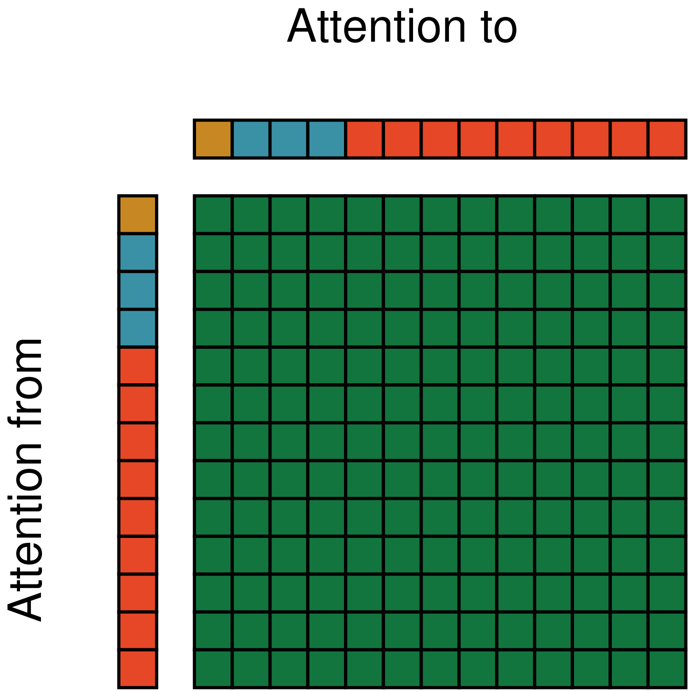
Set-Encoder

Attention Mechanism

[CLS] learn python [SEP] Python is a great language to learn . [SEP]

[CLS] learn python [SEP] Pythons live in the rainforest . [SEP]

[CLS] learn python [SEP] Guido van Rossum invented Python . [SEP]



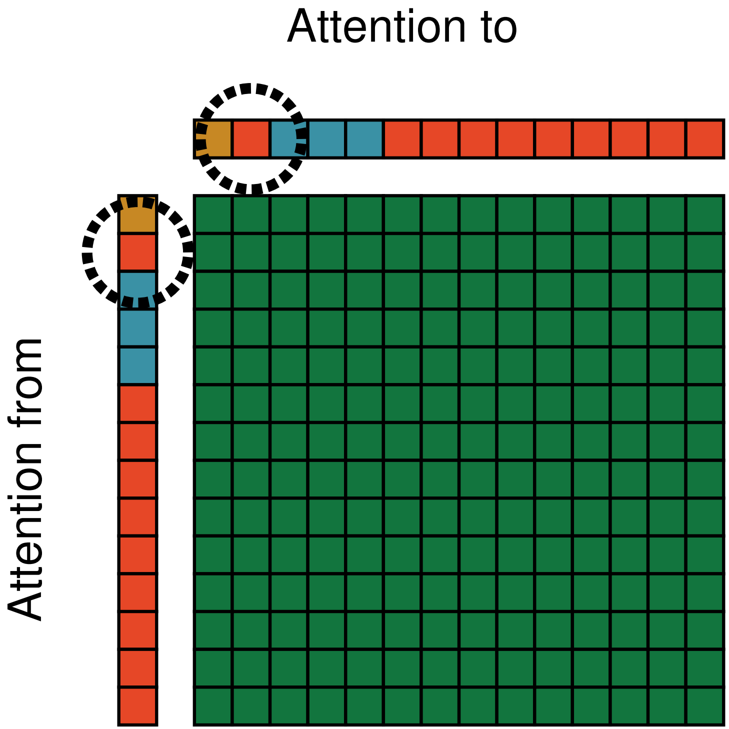
Set-Encoder

Attention Mechanism

[CLS] [INT] learn python [SEP] Python is a great language to learn . [SEP]

[CLS] [INT] learn python [SEP] Pythons live in the rainforest . [SEP]

[CLS] [INT] learn python [SEP] Guido van Rossum invented Python . [SEP]



1. Insert an extra [INT] token

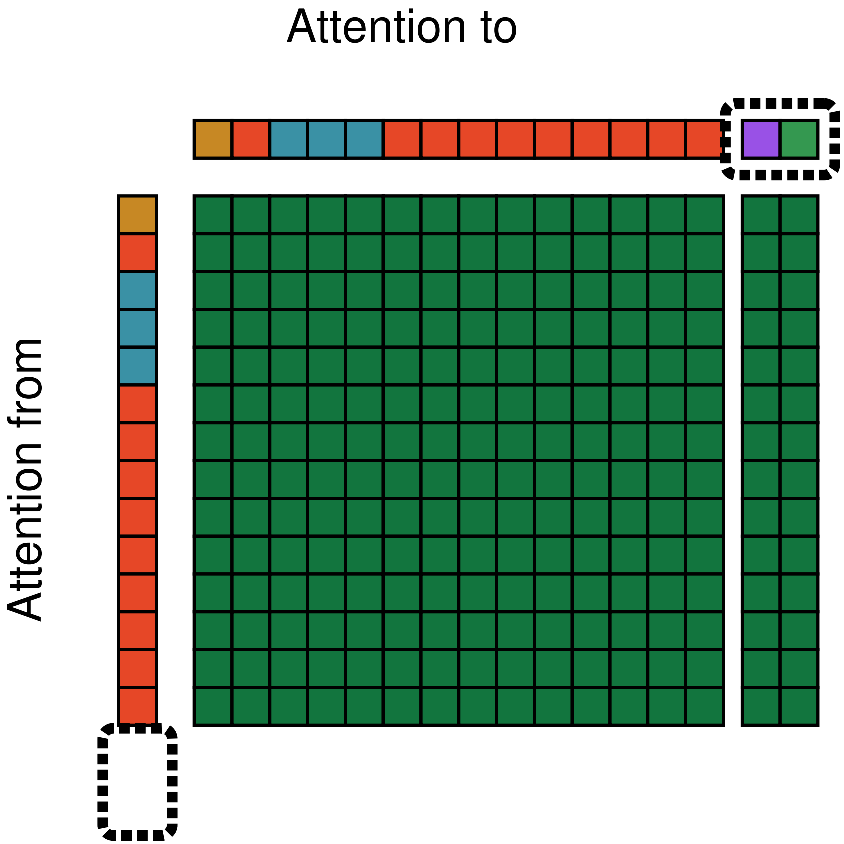
Set-Encoder

Attention Mechanism

[CLS] [INT] learn python [SEP] Python is a great language to learn . [SEP]

[CLS] [INT] learn python [SEP] Pythons live in the rainforest . [SEP]

[CLS] [INT] learn python [SEP] Guido van Rossum invented Python . [SEP]



1. Insert an extra [INT] token
2. Allow a document to attend to all other documents' [INT] tokens

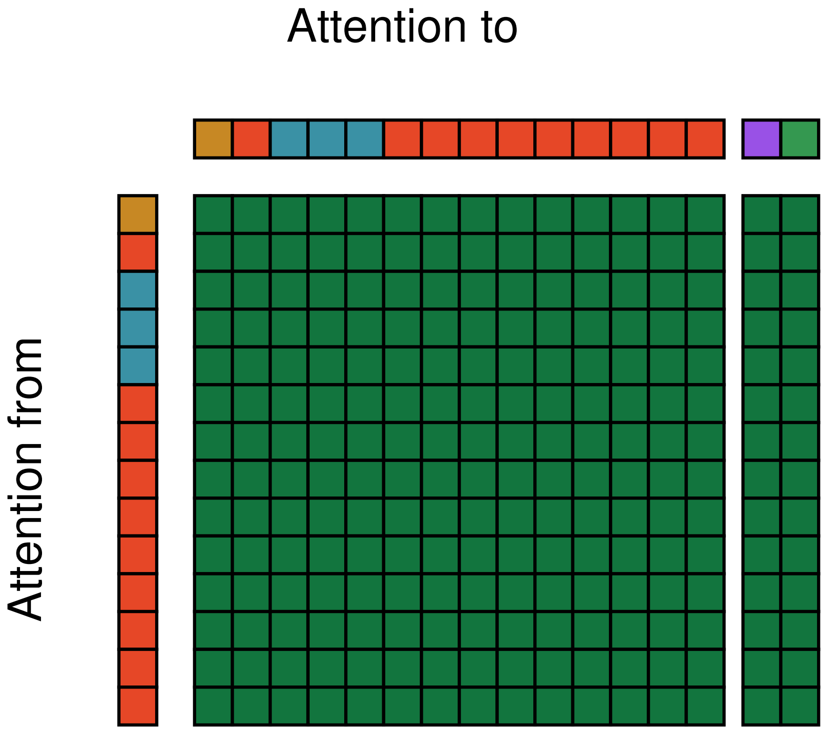
Set-Encoder

Attention Mechanism

[CLS] [INT] learn python [SEP] Python is a great language to learn . [SEP]

[CLS] [INT] learn python [SEP] Pythons live in the rainforest . [SEP]

[CLS] [INT] learn python [SEP] Guido van Rossum invented Python . [SEP]



1. Insert an extra [INT] token
2. Allow a document to attend to all other documents' [INT] tokens

❑ [INT] tokens aggregate semantic information and shares information with other documents

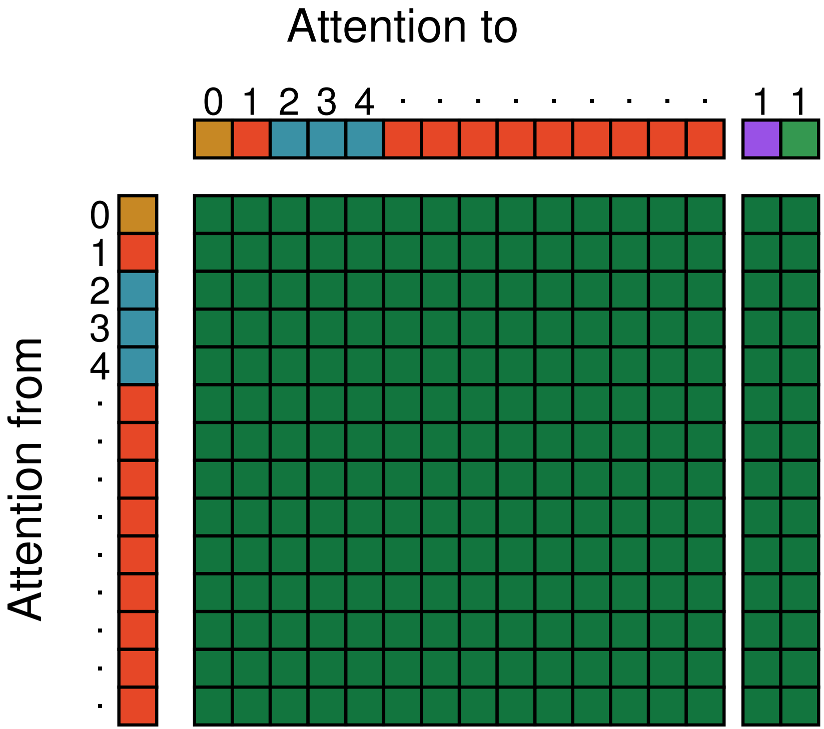
Set-Encoder

Attention Mechanism

[CLS] [INT] learn python [SEP] Python is a great language to learn . [SEP]

[CLS] [INT] learn python [SEP] Pythons live in the rainforest . [SEP]

[CLS] [INT] learn python [SEP] Guido van Rossum invented Python . [SEP]



1. Insert an extra [INT] token
2. Allow a document to attend to all other documents' [INT] tokens

- ❑ [INT] tokens aggregate semantic information and shares information with other documents
- ❑ Permutation-invariant because all [INT] tokens share the same positional encoding

Set-Encoder

Effectiveness

nDCG@10 on TREC Deep Learning 2019 and 2020 passage and TIREx

Model	**DL’19**	**DL’20**	**TIREx**
BM25	0.480	0.494	0.286
monoT5 3B	0.705	0.715	0.313
RankT5 3B	0.710	0.711	**0.322**
RankGPT-4o	0.725	0.719	–
RankZephyr	0.719	0.720	0.320
Set-Encoder$_{\texttt{BASE}}$			
Set-Encoder$_{\texttt{LARGE}}$			

Set-Encoder

Effectiveness

nDCG@10 on TREC Deep Learning 2019 and 2020 passage and TIREx

Model	**DL'19**	**DL'20**	**TIREx**
BM25	0.480	0.494	0.286
monoT5 3B	0.705	0.715	0.313
RankT5 3B	0.710	0.711	**0.322**
RankGPT-4o	0.725	0.719	–
RankZephyr	0.719	0.720	0.320
Set-Encoder$_{\texttt{BASE}}$	0.724	0.710	0.311
Set-Encoder$_{\texttt{LARGE}}$			

❑ Set-Encoder is on-par with SOTA re-rankers in-domain

Set-Encoder

Effectiveness

nDCG@10 on TREC Deep Learning 2019 and 2020 passage and TIREx

Model	**DL'19**	**DL'20**	**TIREx**
BM25	0.480	0.494	0.286
monoT5 3B	0.705	0.715	0.313
RankT5 3B	0.710	0.711	**0.322**
RankGPT-4o	0.725	0.719	–
RankZephyr	0.719	0.720	0.320
Set-Encoder$_{\texttt{BASE}}$	0.724	0.710	0.311
Set-Encoder$_{\texttt{LARGE}}$	**0.727**	**0.735**	0.321

❑ Set-Encoder is on-par with SOTA re-rankers in-domain and out-of-domain

Set-Encoder

Effectiveness

nDCG@10 on TREC Deep Learning 2019 and 2020 passage and TIREx

Model	**DL’19**	**DL’20**	**TIREx**
BM25	0.480	0.494	0.286
monoT5 3B	0.705	0.715	0.313
RankT5 3B	0.710	0.711	**0.322**
RankGPT-4o	0.725	0.719	–
RankZephyr	0.719	0.720	0.320
Set-Encoder$_{\texttt{BASE}}$	0.724	0.710	0.311
Set-Encoder$_{\texttt{LARGE}}$	**0.727**	**0.735**	0.321

- Set-Encoder is on-par with SOTA re-rankers in-domain and out-of-domain
- Despite being distilled from RankZephyr, the Set-Encoder is more effective

Set-Encoder

Effectiveness

nDCG@10 on TREC Deep Learning 2019 and 2020 passage and TIREx

Model	**DL'19**	**DL'20**	**TIREx**
BM25	0.480	0.494	0.286
monoT5 3B	0.705	0.715	0.313
RankT5 3B	0.710	0.711	**0.322**
RankGPT-4o	0.725	0.719	–
RankZephyr	0.719	0.720	0.320
Set-Encoder$_{\texttt{BASE}}$	0.724	0.710	0.311
Set-Encoder$_{\texttt{LARGE}}$	**0.727**	**0.735**	0.321

❑ Set-Encoder is on-par with SOTA re-rankers in-domain and out-of-domain

❑ Despite being distilled from RankZephyr, the Set-Encoder is more effective

➜ LLM-rankers are not permutation-invariant and affected by the first-stage

Set-Encoder

Permutation Invariance

Re-ordering input documents affects previous listwise model’s ranking preferences.

Set-Encoder

Permutation Invariance

Re-ordering input documents affects previous listwise model’s ranking preferences.

We create corrupted BM25 rankings to test a model’s robustness to permutations.

1. Inverse ideal ranking
2. Randomly shuffled ranking
3. Original BM25 ranking
4. Ideal ranking

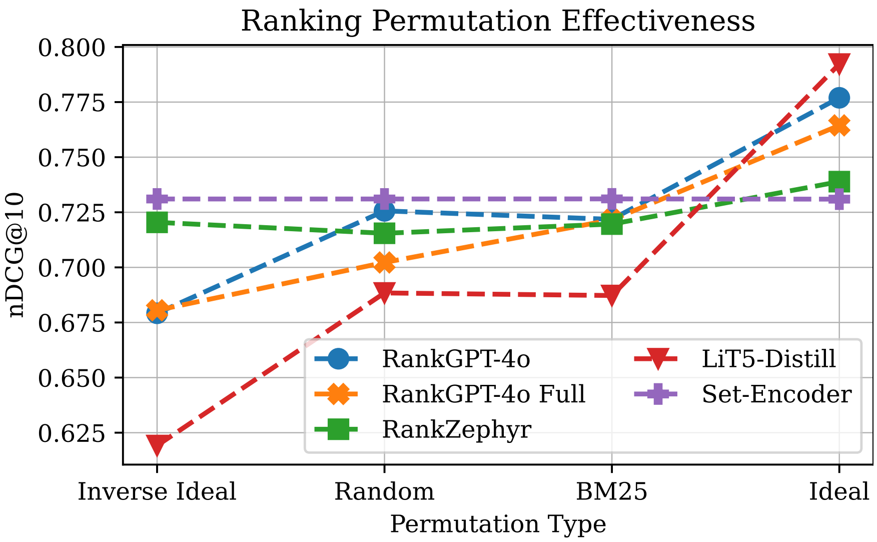
Set-Encoder

Permutation Invariance

Re-ordering input documents affects previous listwise model's ranking preferences.

We create corrupted BM25 rankings to test a model's robustness to permutations.

1. Inverse ideal ranking
2. Randomly shuffled ranking
3. Original BM25 ranking
4. Ideal ranking



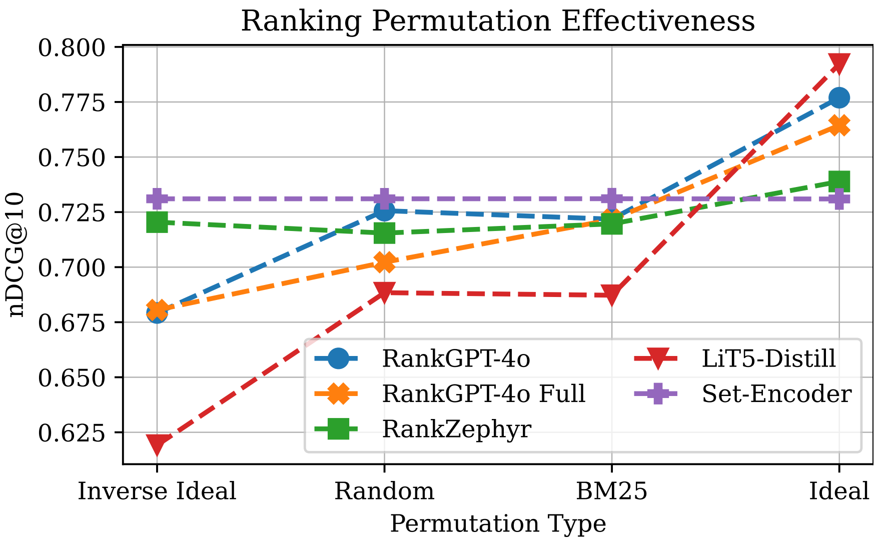
Set-Encoder

Permutation Invariance

Re-ordering input documents affects previous listwise model's ranking preferences.

We create corrupted BM25 rankings to test a model's robustness to permutations.

1. Inverse ideal ranking
2. Randomly shuffled ranking
3. Original BM25 ranking
4. Ideal ranking



❑ Previous listwise re-rankers are biased by the order of the input documents

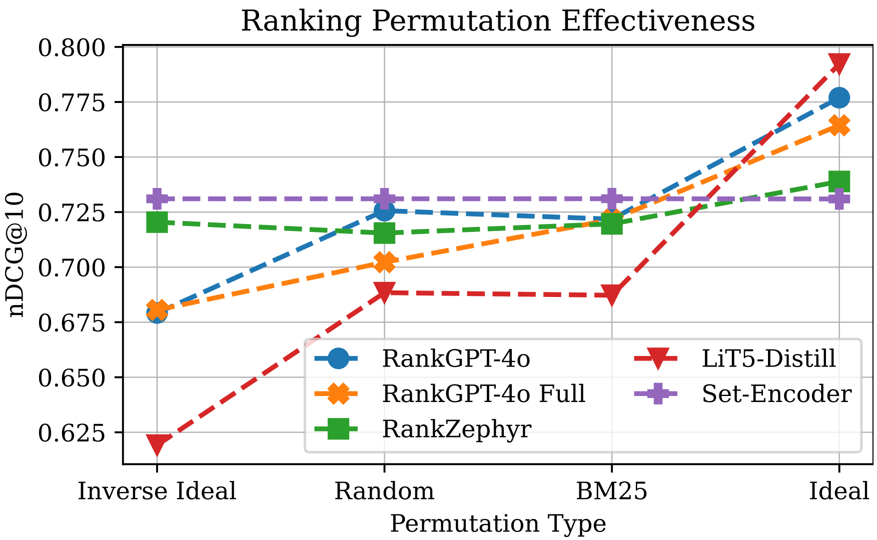
Set-Encoder

Permutation Invariance

Re-ordering input documents affects previous listwise model's ranking preferences.

We create corrupted BM25 rankings to test a model's robustness to permutations.

1. Inverse ideal ranking
2. Randomly shuffled ranking
3. Original BM25 ranking
4. Ideal ranking



- ❑ Previous listwise re-rankers are biased by the order of the input documents
- ❑ Set-Encoder is invariant to the order of the input documents

Set-Encoder

Effectiveness

What about comparing the Set-Encoder to a standard pointwise cross-encoder?

Set-Encoder

Effectiveness

What about comparing the Set-Encoder to a standard pointwise cross-encoder?

nDCG@10 on TREC Deep Learning 2019 and 2020 passage and TIREx

Model	**TREC DL 19**	**TREC DL 20**	**TIREx**
Set-Encoder$_{\texttt{BASE}}$	0.724	0.710	0.311
Set-Encoder$_{\texttt{LARGE}}$	0.727	0.735	0.321
monoELECTRA$_{\texttt{BASE}}$			
monoELECTRA$_{\texttt{LARGE}}$			

Set-Encoder

Effectiveness

What about comparing the Set-Encoder to a standard pointwise cross-encoder?

nDCG@10 on TREC Deep Learning 2019 and 2020 passage and TIREx

Model	**TREC DL 19**	**TREC DL 20**	**TIREx**
Set-Encoder$_{\texttt{BASE}}$	0.724	0.710	0.311
Set-Encoder$_{\texttt{LARGE}}$	0.727	0.735	0.321
monoELECTRA$_{\texttt{BASE}}$	0.720	0.711	0.314
monoELECTRA$_{\texttt{LARGE}}$	0.733	0.727	0.321

- Pointwise model is as effective as a listwise model (and LLMs)

Set-Encoder

Effectiveness

What about comparing the Set-Encoder to a standard pointwise cross-encoder?

nDCG@10 on TREC Deep Learning 2019 and 2020 passage and TIREx

Model	TREC DL 19	TREC DL 20	TIREx
Set-Encoder$_{\texttt{BASE}}$	0.724	0.710	0.311
Set-Encoder$_{\texttt{LARGE}}$	0.727	0.735	0.321
monoELECTRA$_{\texttt{BASE}}$	0.720	0.711	0.314
monoELECTRA$_{\texttt{LARGE}}$	0.733	0.727	0.321

- Pointwise model is as effective as a listwise model (and LLMs)
- Are document interactions necessary for independent relevance judgements?

Set-Encoder

Listwise Re-Ranking

We build a synthetic task which requires document interactions.

Set-Encoder

Listwise Re-Ranking

We build a synthetic task which requires document interactions.

MS MARCO contains many lexical near-duplicates.

Set-Encoder

Listwise Re-Ranking

We build a synthetic task which requires document interactions.

MS MARCO contains many lexical near-duplicates. (Not actual MS MARCO data.)

Python is a great language to learn.

Python is a great language to learn now.

Pythons live in the rainforest.

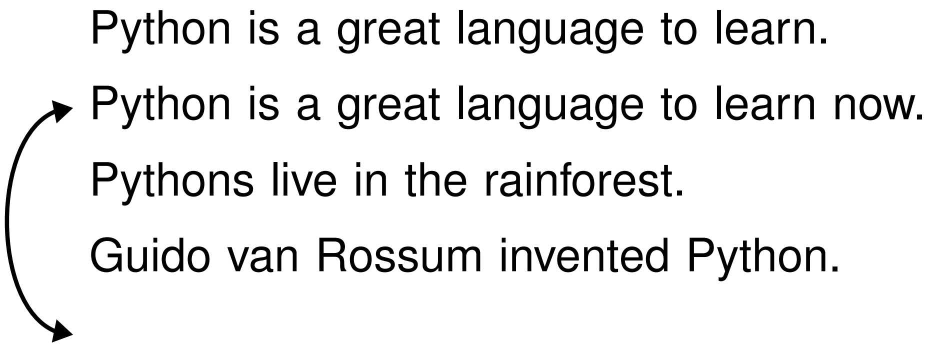
Guido van Rossum invented Python.

Set-Encoder

Listwise Re-Ranking

We build a synthetic task which requires document interactions.

MS MARCO contains many lexical near-duplicates. (Not actual MS MARCO data.)



Python is a great language to learn.

Python is a great language to learn now.

Pythons live in the rainforest.

Guido van Rossum invented Python.

Fine-tune models to rank according to relevance and put duplicates at the end.

Set-Encoder

Listwise Re-Ranking

We build a synthetic task which requires document interactions.

α-nDCG@10 ($\alpha = 0.99$) on the synthetic task

Model	**TREC DL 19**	**TREC DL 20**
First Stage	0.700	0.722
RankGPT-4o	0.741	0.773
RankZephyr	0.700	0.760
monoELECTRA	0.785	0.753
Set-Encoder	**0.821**	**0.803**
Set-Enc. ~~[INT]~~		

Set-Encoder

Listwise Re-Ranking

We build a synthetic task which requires document interactions.

α-nDCG@10 ($\alpha = 0.99$) on the synthetic task

Model	**TREC DL 19**	**TREC DL 20**
First Stage	0.700	0.722
RankGPT-4o	0.741	0.773
RankZephyr	0.700	0.760
monoELECTRA	0.785	0.753
Set-Encoder	**0.821**	**0.803**
Set-Enc. ~~[INT]~~		

- Set-Encoder improves over baselines in novelty-aware re-ranking

Set-Encoder

Listwise Re-Ranking

We build a synthetic task which requires document interactions.

α-nDCG@10 ($\alpha = 0.99$) on the synthetic task

Model	**TREC DL 19**	**TREC DL 20**
First Stage	0.700	0.722
RankGPT-4o	0.741	0.773
RankZephyr	0.700	0.760
monoELECTRA	0.785	0.753
Set-Encoder	**0.821**	**0.803**
Set-Enc. ~~[INT]~~	0.773	0.748

- ❑ Set-Encoder improves over baselines in novelty-aware re-ranking
- ❑ Without the interaction token, the Set-Encoder is less effective

Set-Encoder

Intermediate Conclusion

The Set-Encoder enables permutation-invariant inter-document interactions.

Set-Encoder

Intermediate Conclusion

The Set-Encoder enables permutation-invariant inter-document interactions.

- It is on-par with SOTA LLM re-rankers

Set-Encoder

Intermediate Conclusion

The Set-Encoder enables permutation-invariant inter-document interactions.

- It is on-par with SOTA LLM re-rankers
- Permutation invariance is crucial for robustness and efficiency

Set-Encoder

Intermediate Conclusion

The Set-Encoder enables permutation-invariant inter-document interactions.

- It is on-par with SOTA LLM re-rankers
- Permutation invariance is crucial for robustness and efficiency
- **BUT:** Interactions probably not necessary for independent judgments

Set-Encoder

Intermediate Conclusion

The Set-Encoder enables permutation-invariant inter-document interactions.

- It is on-par with SOTA LLM re-rankers
- Permutation invariance is crucial for robustness and efficiency
- **BUT:** Interactions probably not necessary for independent judgments

➜ More complex tasks are necessary to evaluate modern models

Set-Encoder

Intermediate Conclusion

The Set-Encoder enables permutation-invariant inter-document interactions.

- It is on-par with SOTA LLM re-rankers
- Permutation invariance is crucial for robustness and efficiency
- **BUT:** Interactions probably not necessary for independent judgments

➜ More complex tasks are necessary to evaluate modern models

Checkpoints are released on HF and can be used with our Lightning IR framework.

Lightning IR

Set-Encoder

Intermediate Conclusion

The Set-Encoder enables permutation-invariant inter-document interactions.

- It is on-par with SOTA LLM re-rankers
- Permutation invariance is crucial for robustness and efficiency
- **BUT:** Interactions probably not necessary for independent judgments

➜ More complex tasks are necessary to evaluate modern models

Checkpoints are released on HF and can be used with our Lightning IR framework.

Lightning IR

Code and paper @
webis-de/set-encoder

Set-Encoder

Intermediate Conclusion

The Set-Encoder enables permutation-invariant inter-document interactions.

- ❑ It is on-par with SOTA LLM re-rankers
- ❑ Permutation invariance is crucial for robustness and efficiency
- ❑ **BUT:** Interactions probably not necessary for independent judgments
- ➜ More complex tasks are necessary to evaluate modern models

Checkpoints are released on HF and can be used with our Lightning IR framework.

Lightning IR

Questions?

Code and paper @
webis-de/set-encoder

Token-Independent Text Encoder (TITE)

Standard Bi-Encoder Model

Token-Independent Text Encoder (TITE)

Standard Bi-Encoder Model

Query learn python

Documents Python is a great language to learn.
Pythons live in the rainforest.
Guido van Rossum invented Python.

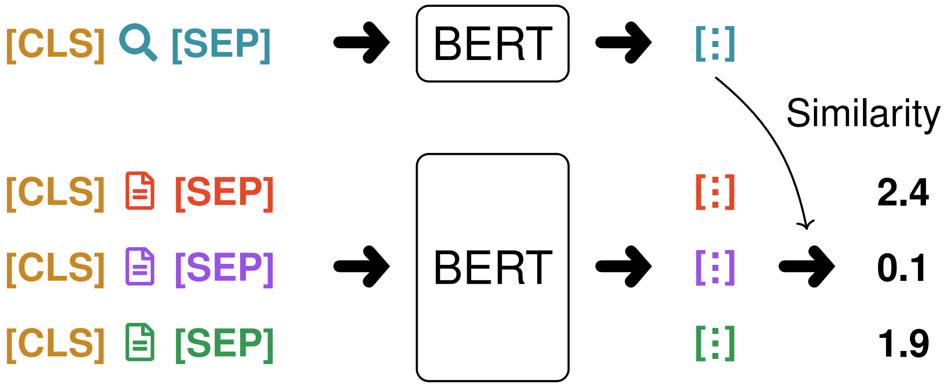
Token-Independent Text Encoder (TITE)

Standard Bi-Encoder Model

Query 🔍 learn python

Documents 📄 Python is a great language to learn.
Pythons live in the rainforest.
Guido van Rossum invented Python.

Sentence-BERT [Reimers and Gurevych, EMNLP’19]



[CLS] 🔍 [SEP] → BERT → [⋮]

Similarity

[CLS] 📄 [SEP]
[CLS] 📄 [SEP] → BERT → [⋮] → 2.4
[CLS] 📄 [SEP]

[⋮] 0.1
[⋮] 1.9

Token-Independent Text Encoder (TITE)

Pooling in Bi-Encoder Models

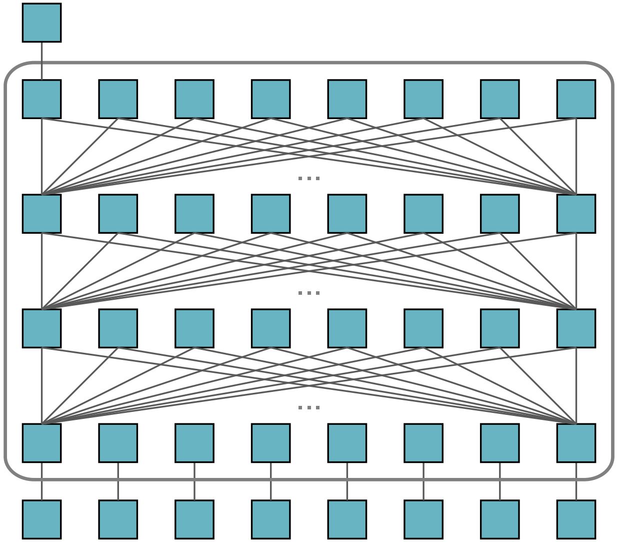
To obtain a single vector, bi-encoder models pool the token representations.

Token-Independent Text Encoder (TITE)

Pooling in Bi-Encoder Models

To obtain a single vector, bi-encoder models pool the token representations.

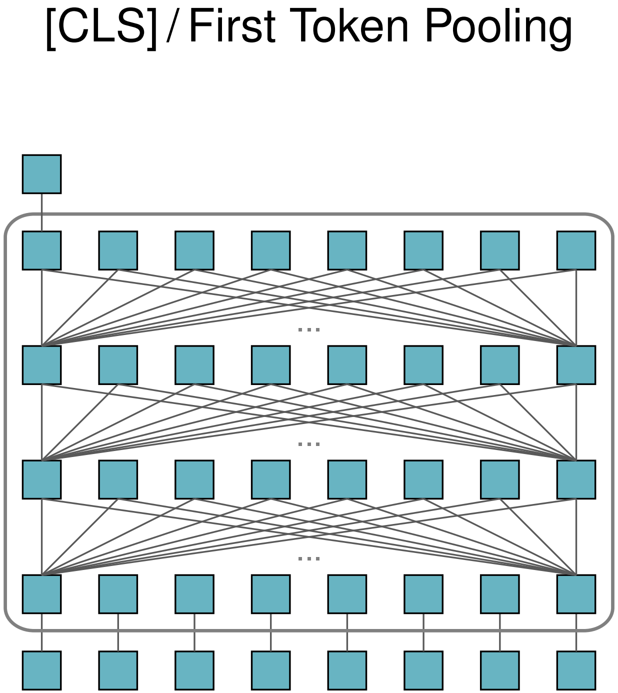
[CLS] / First Token Pooling

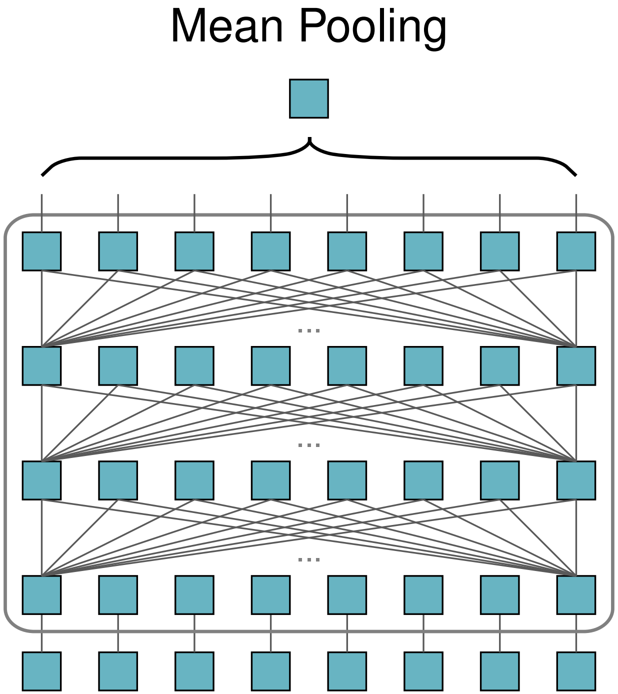


Token-Independent Text Encoder (TITE)

Pooling in Bi-Encoder Models

To obtain a single vector, bi-encoder models pool the token representations.

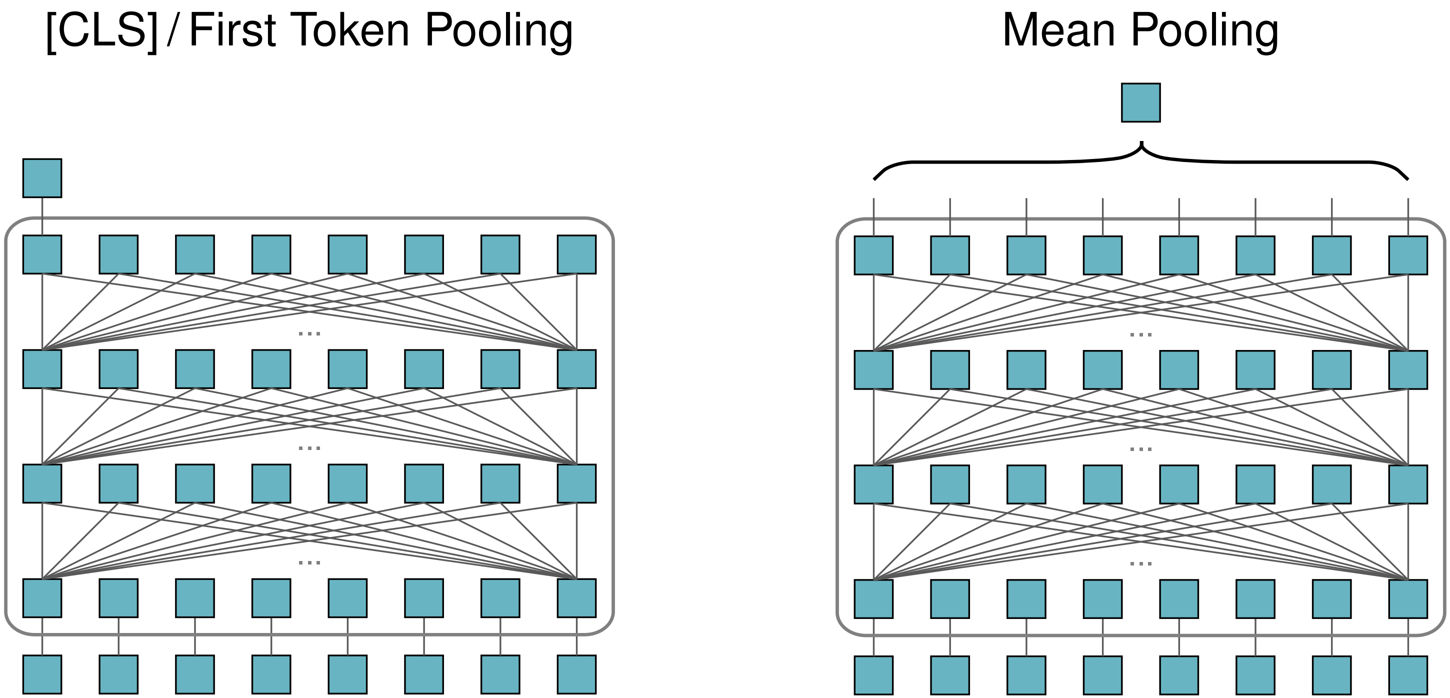




Token-Independent Text Encoder (TITE)

Pooling in Bi-Encoder Models

To obtain a single vector, bi-encoder models pool the token representations.

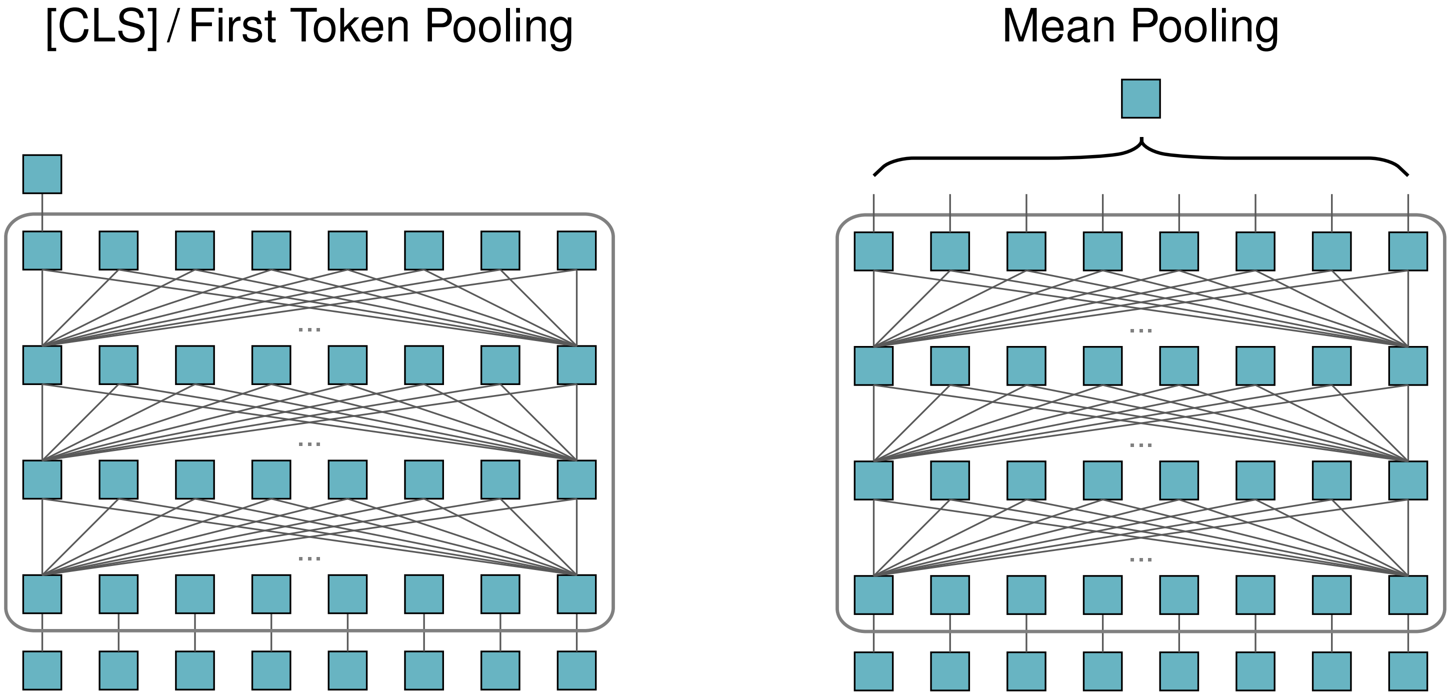


- ❑ [CLS] pooling learns aggregation but discards token representations

Token-Independent Text Encoder (TITE)

Pooling in Bi-Encoder Models

To obtain a single vector, bi-encoder models pool the token representations.



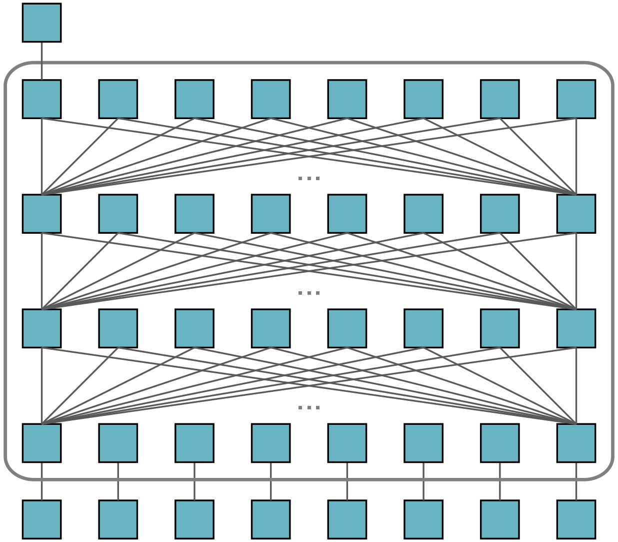
- [CLS] pooling learns aggregation but discards token representations
- Mean pooling is static but uses all token representations

Token-Independent Text Encoder (TITE)

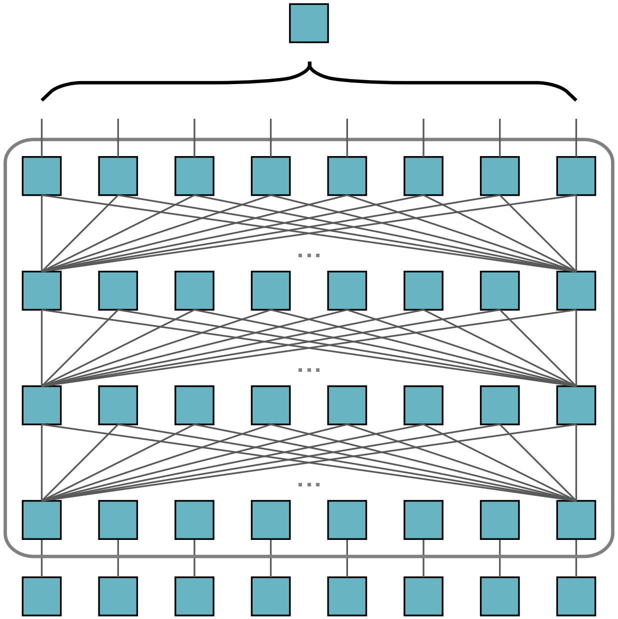
Pooling in Bi-Encoder Models

To obtain a single vector, bi-encoder models pool the token representations.

[CLS] / First Token Pooling



Mean Pooling



❑ [CLS] pooling learns aggregation but discards token representations

❑ Mean pooling is static but uses all token representations

➜ Combine both approaches by pooling within the transformer layers

Token-Independent Text Encoder (TITE)

Encoder Model with Single Vector Output

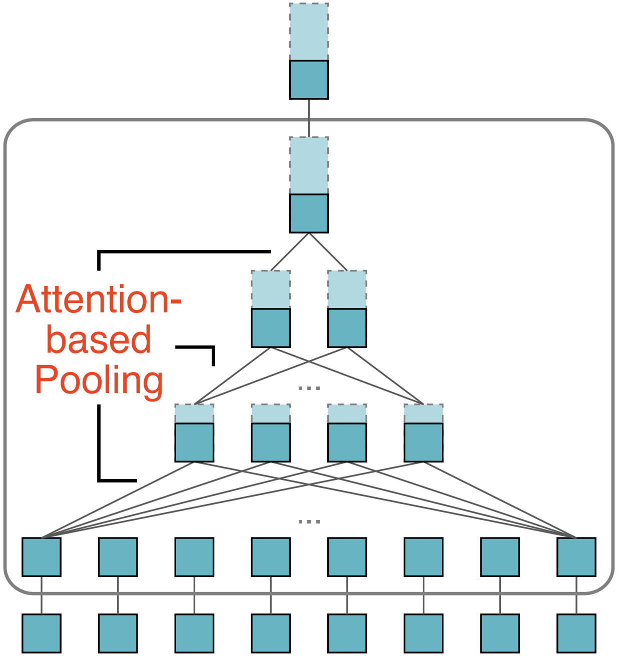
Combine both approaches by pooling within the transformer layers.

Token-Independent Text Encoder (TITE)

Encoder Model with Single Vector Output

Combine both approaches by pooling within the transformer layers.

Token-Independent Text Encoder

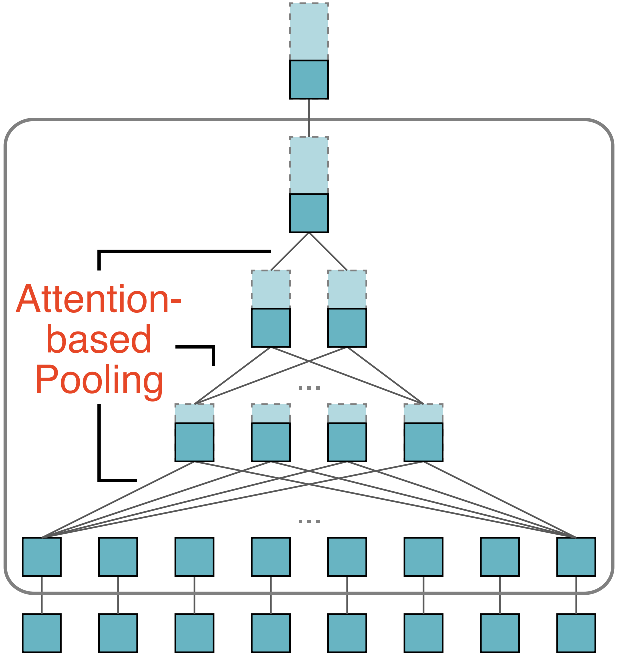


Token-Independent Text Encoder (TITE)

Encoder Model with Single Vector Output

Combine both approaches by pooling within the transformer layers.

Token-Independent Text Encoder



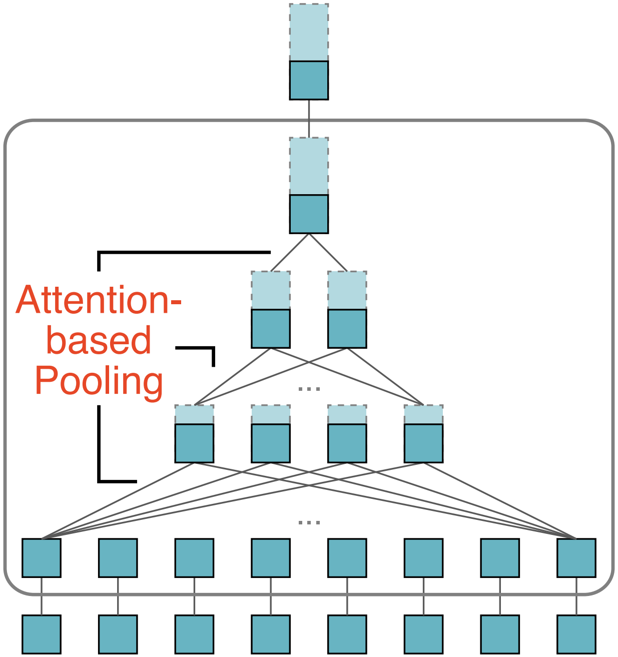
- ❑ TITE outputs a single sequence-level vector for an input sequence

Token-Independent Text Encoder (TITE)

Encoder Model with Single Vector Output

Combine both approaches by pooling within the transformer layers.

Token-Independent Text Encoder



- ❑ TITE outputs a single sequence-level vector for an input sequence
- ❑ Optionally, the dimensionality of vectors can be increased

Token-Independent Text Encoder (TITE)

Attention-based Pooling

Attention-based pooling builds on the Funnel Transformer. [Dai et al., NeurIPS'20]

Token-Independent Text Encoder (TITE)

Attention-based Pooling

Attention-based pooling builds on the Funnel Transformer. [Dai et al., NeurIPS’20]

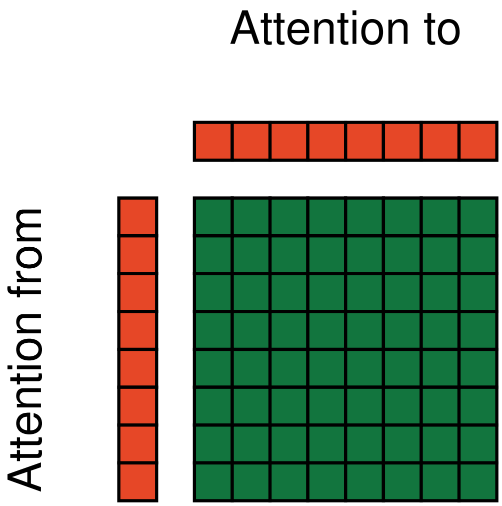
Python is a great language to learn .   (Process is the same for queries)

Token-Independent Text Encoder (TITE)

Attention-based Pooling

Attention-based pooling builds on the Funnel Transformer. [Dai et al., NeurIPS’20]

Python is a great language to learn . (Process is the same for queries)

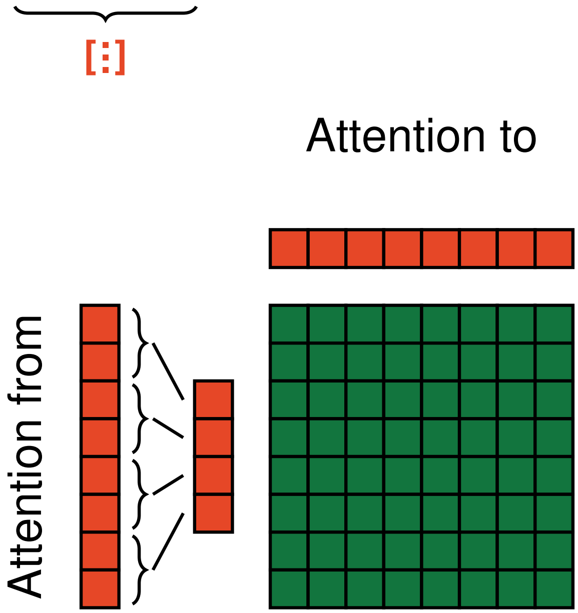


Token-Independent Text Encoder (TITE)

Attention-based Pooling

Attention-based pooling builds on the Funnel Transformer. [Dai et al., NeurIPS’20]

Python is a great language to learn . (Process is the same for queries)



1. Pool the representations of neighboring tokens

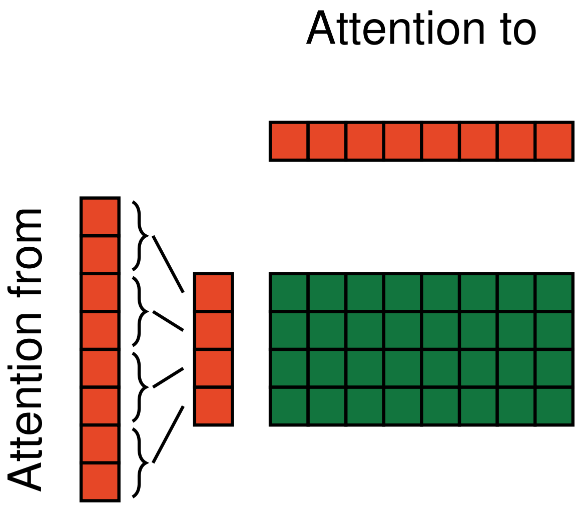
Token-Independent Text Encoder (TITE)

Attention-based Pooling

Attention-based pooling builds on the Funnel Transformer. [Dai et al., NeurIPS'20]

Python is a great language to learn .   (Process is the same for queries)

$\underbrace{\text{Python is}}_{[:]}$



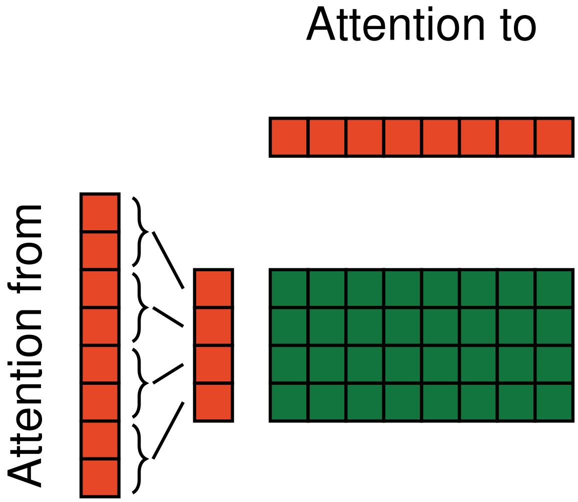
1. Pool the representations of neighboring tokens
2. Allow a pooled "meta-token" to attend to all tokens of the previous layer

Token-Independent Text Encoder (TITE)

Attention-based Pooling

Attention-based pooling builds on the Funnel Transformer. [Dai et al., NeurIPS’20]

$\underbrace{\text{Python is a}}_{[\vdots]}$ great language to learn .   (Process is the same for queries)



1. Pool the representations of neighboring tokens
2. Allow a pooled “meta-token” to attend to all tokens of the previous layer

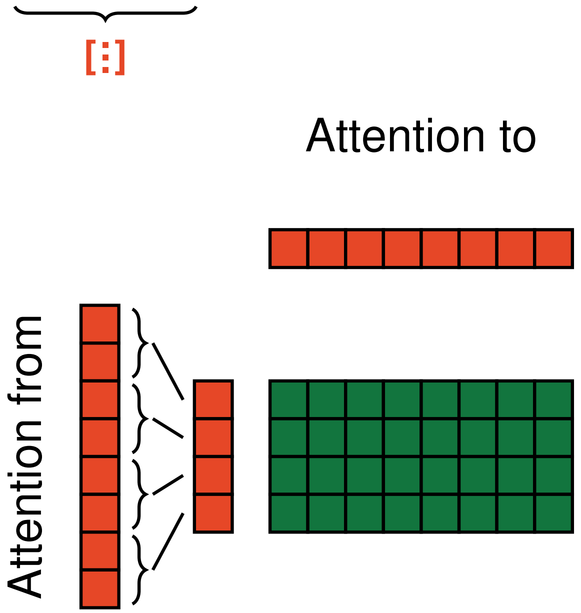
❑ The output representations become smaller with each layer

Token-Independent Text Encoder (TITE)

Attention-based Pooling

Attention-based pooling builds on the Funnel Transformer. [Dai et al., NeurIPS’20]

Python is a great language to learn . (Process is the same for queries)



1. Pool the representations of neighboring tokens

2. Allow a pooled “meta-token” to attend to all tokens of the previous layer

- The output representations become smaller with each layer

- Fine-grained attention across “meta-tokens”

Token-Independent Text Encoder (TITE)

Efficiency and Effectiveness

Token-Independent Text Encoder (TITE)

Efficiency and Effectiveness

Efficiency

Model	**Queries**	**Docs**
BERT	48.0	8.7
ModernBERT		
TITE (Base)		
TITE (Upscale)		

Queries and documents per second (×1,000)

Token-Independent Text Encoder (TITE)

Efficiency and Effectiveness

Efficiency

Model	Queries	Docs
BERT	48.0	8.7
ModernBERT		
TITE (Base)	89.0 (1.9×)	20.8 (2.4×)
TITE (Upscale)		

Queries and documents per second (×1,000)

- Around 2× faster than BERT

Token-Independent Text Encoder (TITE)

Efficiency and Effectiveness

Efficiency

Model	**Queries**	**Docs**
BERT	48.0	8.7
ModernBERT		
TITE (Base)	89.0 (1.9×)	20.8 (2.4×)
TITE (Upscale)	70.1 (1.5×)	16.3 (1.9×)

Queries and documents per second (×1,000)

- Around 2× faster than BERT
- Upscaling incurs small overhead

Token-Independent Text Encoder (TITE)

Efficiency and Effectiveness

Efficiency

Model	Queries	Docs
BERT	48.0	8.7
ModernBERT	41.1 (0.9×)	8.3 (1.0×)
TITE (Base)	89.0 (1.9×)	20.8 (2.4×)
TITE (Upscale)	70.1 (1.5×)	16.3 (1.9×)

Queries and documents per second (×1,000)

- Around 2× faster than BERT
- Upscaling incurs small overhead
- Flash BERT > ModernBERT

Token-Independent Text Encoder (TITE)

Efficiency and Effectiveness

Efficiency

Model	Queries	Docs
BERT	48.0	8.7
ModernBERT	41.1 (0.9×)	8.3 (1.0×)
TITE (Base)	89.0 (1.9×)	20.8 (2.4×)
TITE (Upscale)	70.1 (1.5×)	16.3 (1.9×)

Queries and documents per second (×1,000)

Effectiveness

Model	DL'19	DL'20	BEIR
S-BERT	.700	.688	.449
RetroMAE			
TITE (Base)			
TITE (Upscale)			

nDCG@10 on TREC DL and BEIR

- Around 2× faster than BERT
- Upscaling incurs small overhead
- Flash BERT > ModernBERT

Token-Independent Text Encoder (TITE)

Efficiency and Effectiveness

Efficiency

Model	Queries	Docs
BERT	48.0	8.7
ModernBERT	41.1 (0.9×)	8.3 (1.0×)
TITE (Base)	89.0 (1.9×)	20.8 (2.4×)
TITE (Upscale)	70.1 (1.5×)	16.3 (1.9×)

Queries and documents per second (×1,000)

- Around 2× faster than BERT
- Upscaling incurs small overhead
- Flash BERT > ModernBERT

Effectiveness

Model	DL’19	DL’20	BEIR
S-BERT	.700	.688	.449
RetroMAE			
TITE (Base)	.705	.670	.449
TITE (Upscale)			

nDCG@10 on TREC DL and BEIR

- TITE as effective as S-BERT

Token-Independent Text Encoder (TITE)

Efficiency and Effectiveness

Efficiency

Model	Queries	Docs
BERT	48.0	8.7
ModernBERT	41.1 (0.9×)	8.3 (1.0×)
TITE (Base)	89.0 (1.9×)	20.8 (2.4×)
TITE (Upscale)	70.1 (1.5×)	16.3 (1.9×)

Queries and documents per second (×1,000)

- Around 2× faster than BERT
- Upscaling incurs small overhead
- Flash BERT > ModernBERT

Effectiveness

Model	DL'19	DL'20	BEIR
S-BERT	.700	.688	.449
RetroMAE			
TITE (Base)	.705	.670	.449
TITE (Upscale)	.724	.686	.451

nDCG@10 on TREC DL and BEIR

- TITE as effective as S-BERT
- Upscaling improves effectiveness

Token-Independent Text Encoder (TITE)

Efficiency and Effectiveness

Efficiency

Model	Queries	Docs
BERT	48.0	8.7
ModernBERT	41.1 (0.9×)	8.3 (1.0×)
TITE (Base)	89.0 (1.9×)	20.8 (2.4×)
TITE (Upscale)	70.1 (1.5×)	16.3 (1.9×)

Queries and documents per second (×1,000)

- Around 2× faster than BERT
- Upscaling incurs small overhead
- Flash BERT > ModernBERT

Effectiveness

Model	DL'19	DL'20	BEIR
S-BERT	.700	.688	.449
RetroMAE	.723	.711	.476
TITE (Base)	.705	.670	.449
TITE (Upscale)	.724	.686	.451

nDCG@10 on TREC DL and BEIR

- TITE as effective as S-BERT
- Upscaling improves effectiveness
- Not as effective as RetroMAE

Token-Independent Text Encoder (TITE)

Efficiency and Effectiveness

Efficiency

Model	Queries	Docs
BERT	48.0	8.7
ModernBERT	41.1 (0.9×)	8.3 (1.0×)
TITE (Base)	89.0 (1.9×)	20.8 (2.4×)
TITE (Upscale)	70.1 (1.5×)	16.3 (1.9×)

Queries and documents per second (×1,000)

- Around 2× faster than BERT
- Upscaling incurs small overhead
- Flash BERT > ModernBERT

Effectiveness

Model	DL'19	DL'20	BEIR
S-BERT	.700	.688	.449
RetroMAE	.723	.711	.476
TITE (Base)	.705	.670	.449
TITE (Upscale)	.724	.686	.451

nDCG@10 on TREC DL and BEIR

- TITE as effective as S-BERT
- Upscaling improves effectiveness
- Not as effective as RetroMAE

RetroMAE is a single-vector model pre-trained for text retrieval.

Token-Independent Text Encoder (TITE)

Efficiency and Effectiveness

Efficiency

Model	Queries	Docs
BERT	48.0	8.7
ModernBERT	41.1 (0.9×)	8.3 (1.0×)
TITE (Base)	89.0 (1.9×)	20.8 (2.4×)
TITE (Upscale)	70.1 (1.5×)	16.3 (1.9×)

Queries and documents per second (×1,000)

- Around 2× faster than BERT
- Upscaling incurs small overhead
- Flash BERT > ModernBERT

Effectiveness

Model	DL'19	DL'20	BEIR
S-BERT	.700	.688	.449
RetroMAE	.723	.711	.476
TITE (Base)	.705	.670	.449
TITE (Upscale)	.724	.686	.451

nDCG@10 on TREC DL and BEIR

- TITE as effective as S-BERT
- Upscaling improves effectiveness
- Not as effective as RetroMAE

RetroMAE is a single-vector model pre-trained for text retrieval.

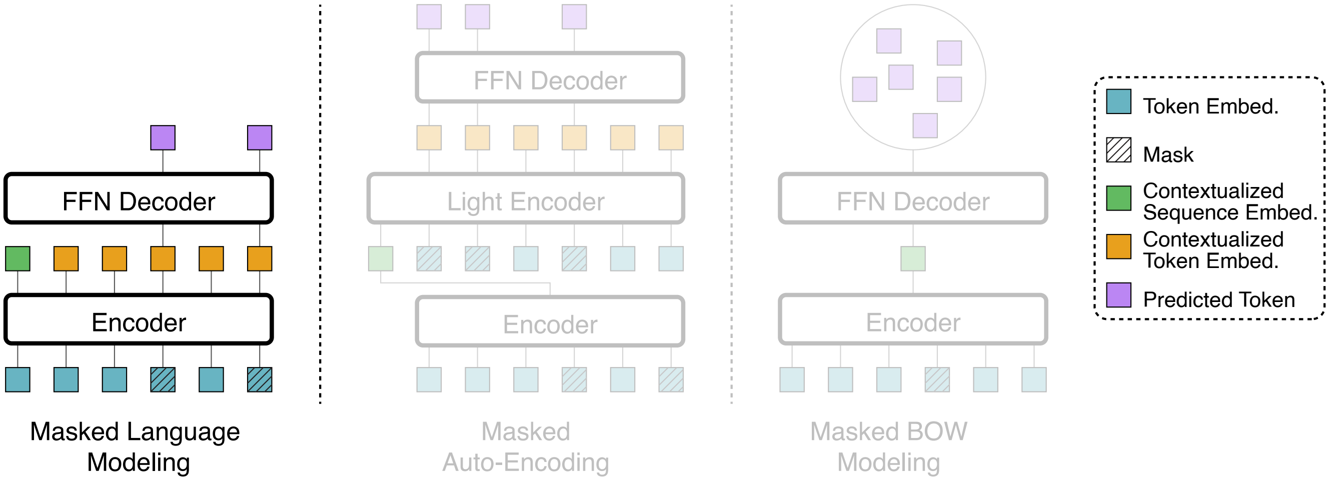
Is TITE less effective than RetroMAE due to the architecture or pre-training?

Token-Independent Text Encoder (TITE)

Sequence-Level Pre-Training

Token-Independent Text Encoder (TITE)

Sequence-Level Pre-Training



FFN Decoder

Encoder

Masked Language Modeling

FFN Decoder

Light Encoder

Encoder

Masked Auto-Encoding

FFN Decoder

Encoder

Masked BOW Modeling

🟦 Token Embed.

▨ Mask

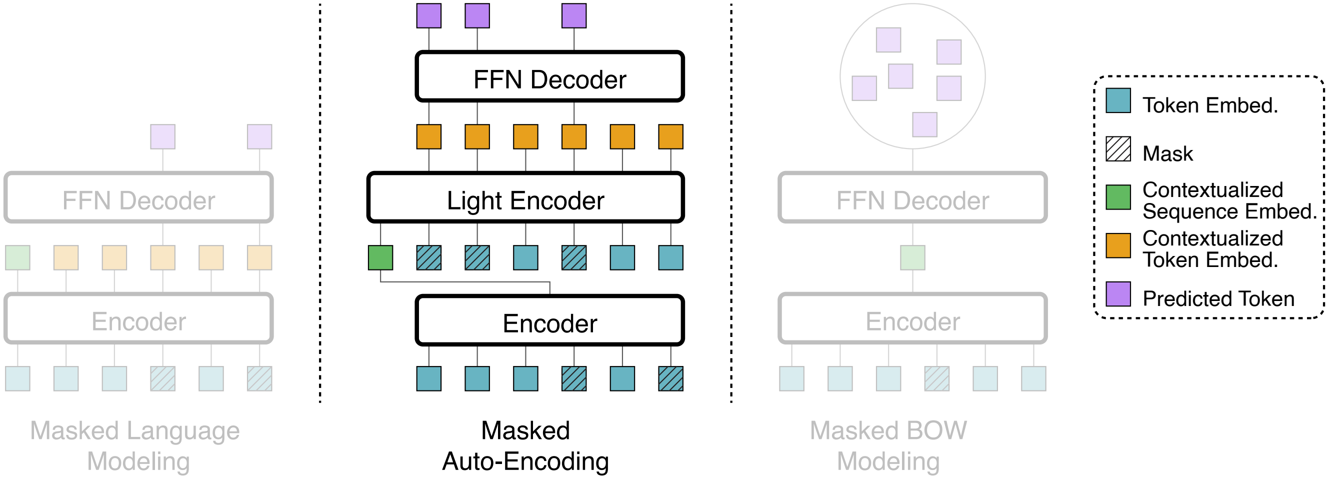
🟩 Contextualized Sequence Embed.

🟧 Contextualized Token Embed.

🟪 Predicted Token

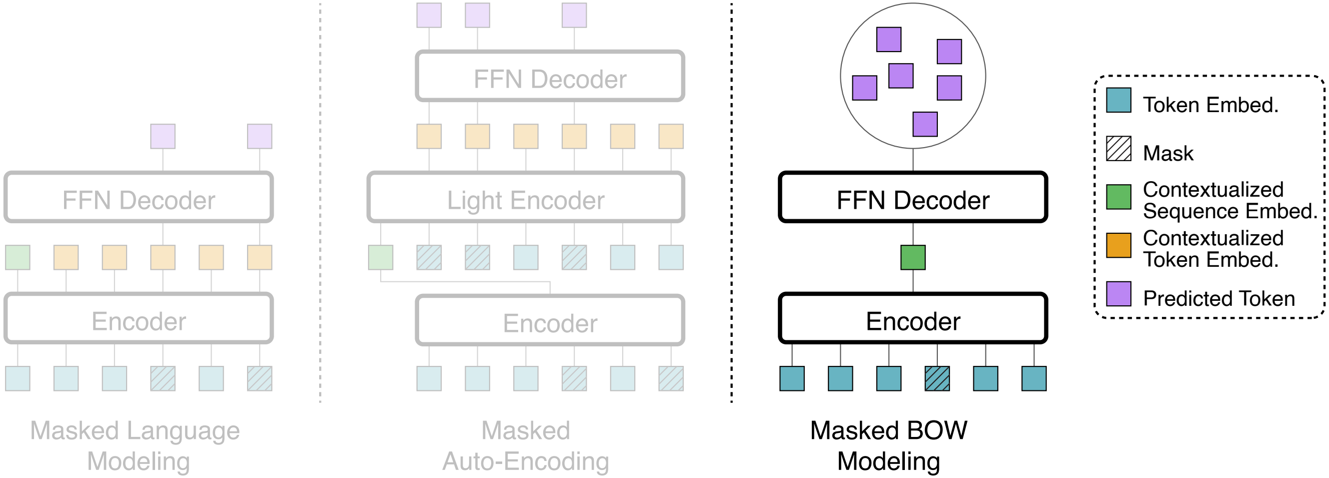
Token-Independent Text Encoder (TITE)

Sequence-Level Pre-Training



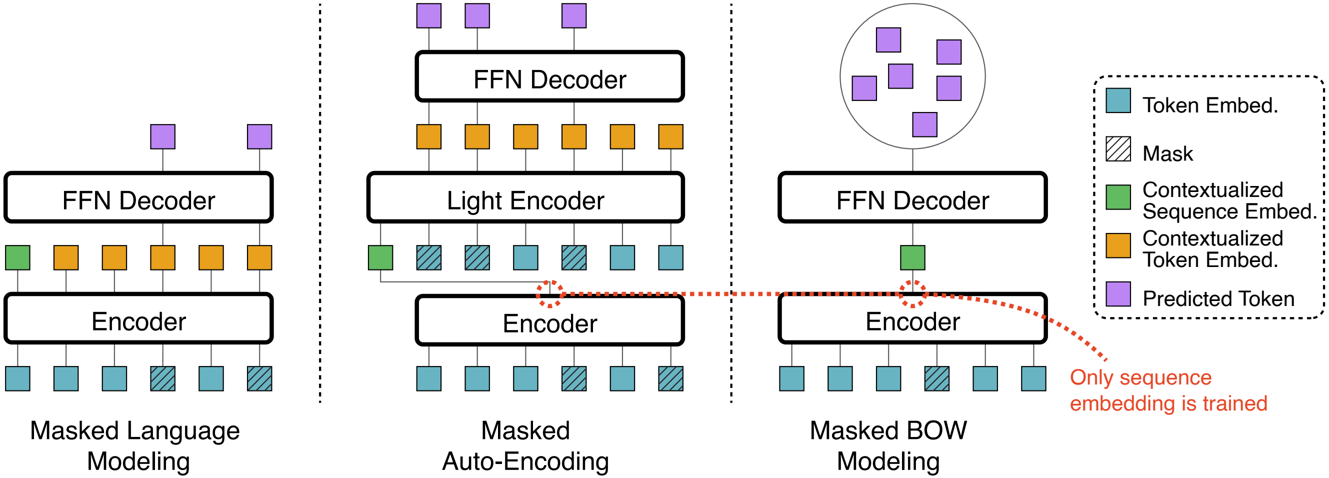
Token-Independent Text Encoder (TITE)

Sequence-Level Pre-Training



Token-Independent Text Encoder (TITE)

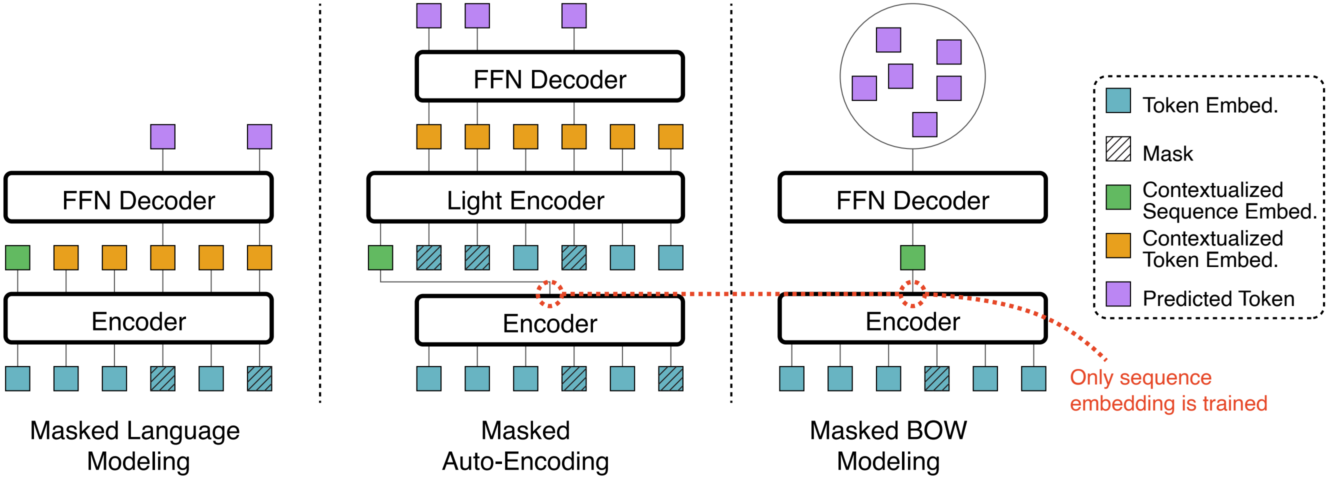
Sequence-Level Pre-Training



Only sequence embedding is trained

Token-Independent Text Encoder (TITE)

Sequence-Level Pre-Training



FFN Decoder

Encoder

Masked Language Modeling

FFN Decoder

Light Encoder

Encoder

Masked Auto-Encoding

FFN Decoder

Encoder

Masked BOW Modeling

Only sequence embedding is trained

🟦 Token Embed.

▨ Mask

🟩 Contextualized Sequence Embed.

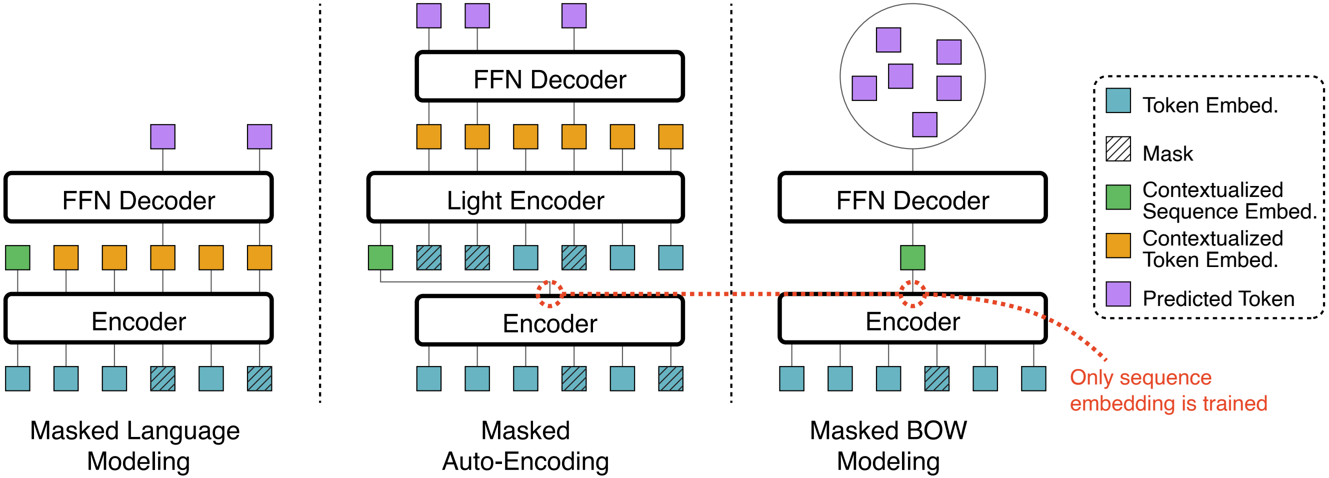
🟧 Contextualized Token Embed.

🟪 Predicted Token

Model	$\mathcal{L}$			DL’19	DL’20	BEIR
	MLM	MAE	BOW			
S-BERT	✓	✗	✗			
RetroMAE	✓	✓	✗			
CLS-BERT	✗	✓	✓			
TITE	✗	✓	✓			
TITE	✗	✓	✗			
TITE	✗	✗	✓			

Token-Independent Text Encoder (TITE)

Sequence-Level Pre-Training

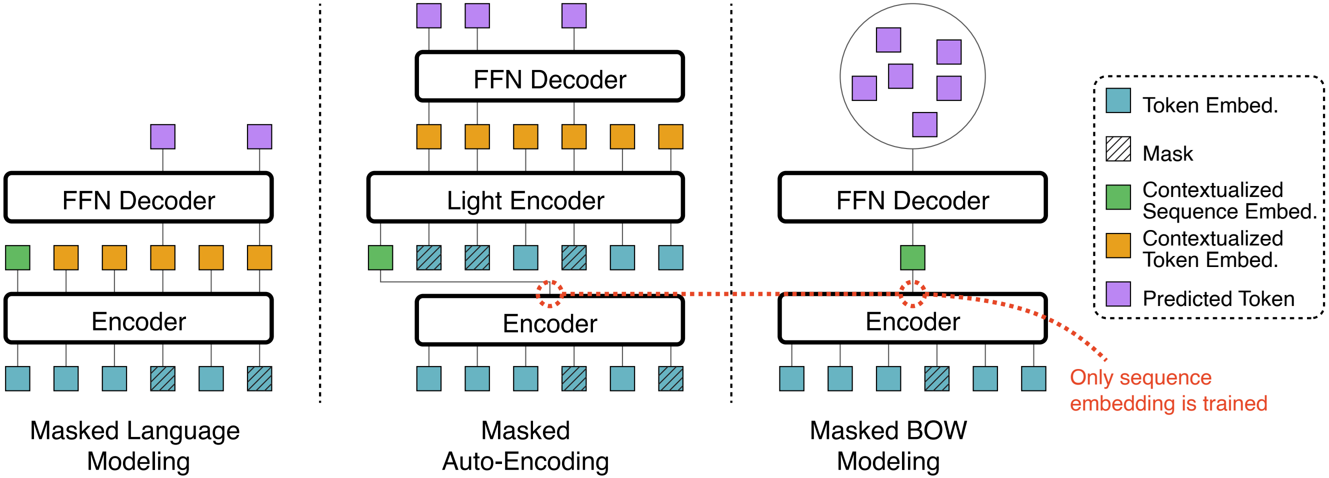


Model	$\mathcal{L}$			DL'19	DL'20	BEIR
	MLM	MAE	BOW			
S-BERT	✓	✗	✗	.700	.688	.449
RetroMAE	✓	✓	✗	.723	.711	.476
CLS-BERT	✗	✓	✓			
TITE	✗	✓	✓			
TITE	✗	✓	✗			
TITE	✗	✗	✓			

❑ MLM + MAE > MLM

Token-Independent Text Encoder (TITE)

Sequence-Level Pre-Training



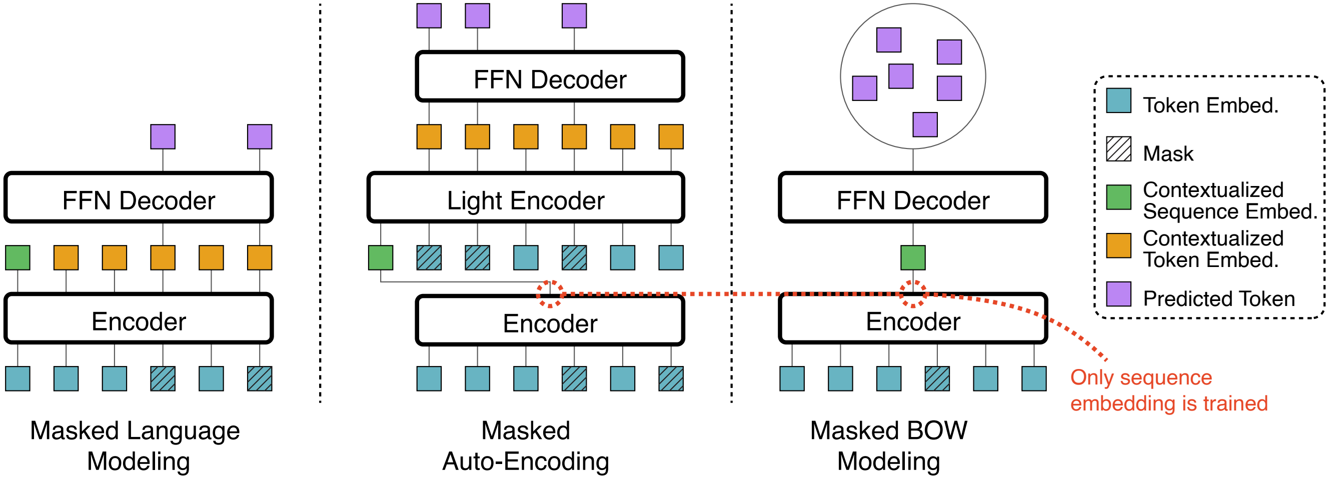
Only sequence embedding is trained

Model	$\mathcal{L}$			**DL'19**	**DL'20**	**BEIR**
	MLM	MAE	BOW			
S-BERT	✓	✗	✗	.700	.688	.449
RetroMAE	✓	✓	✗	.723	.711	.476
CLS-BERT	✗	✓	✓	.704	.674	.444
TITE	✗	✓	✓	.705	.670	.449
TITE	✗	✓	✗			
TITE	✗	✗	✓			

- ❑ $\text{MLM} + \text{MAE} > \text{MLM}$
- ❑ $\text{MLM} \approx \text{MAE} + \text{BOW}$

Token-Independent Text Encoder (TITE)

Sequence-Level Pre-Training

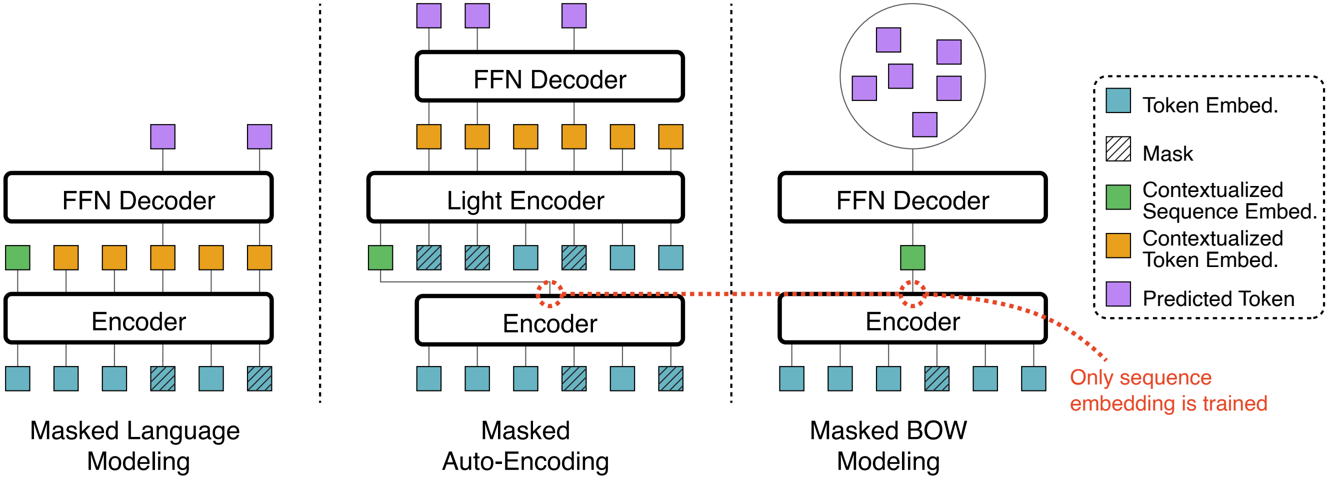


Model	$\mathcal{L}$			DL'19	DL'20	BEIR
	MLM	MAE	BOW			
S-BERT	✓	✗	✗	.700	.688	.449
RetroMAE	✓	✓	✗	.723	.711	.476
CLS-BERT	✗	✓	✓	.704	.674	.444
TITE	✗	✓	✓	.705	.670	.449
TITE	✗	✓	✗	.657	.657	.400
TITE	✗	✗	✓	.660	.676	.426

- $\text{MLM} + \text{MAE} > \text{MLM}$
- $\text{MLM} \approx \text{MAE} + \text{BOW}$
- $\text{MAE} + \text{BOW} > \text{BOW} > \text{MAE}$

Token-Independent Text Encoder (TITE)

Sequence-Level Pre-Training



Model	$\mathcal{L}$			DL’19	DL’20	BEIR
	MLM	MAE	BOW			
S-BERT	✓	✗	✗	.700	.688	.449
RetroMAE	✓	✓	✗	.723	.711	.476
CLS-BERT	✗	✓	✓	.704	.674	.444
TITE	✗	✓	✓	.705	.670	.449
TITE	✗	✓	✗	.657	.657	.400
TITE	✗	✗	✓	.660	.676	.426

❑ $\text{MLM} + \text{MAE} > \text{MLM}$

❑ $\text{MLM} \approx \text{MAE} + \text{BOW}$

❑ $\text{MAE} + \text{BOW} > \text{BOW} > \text{MAE}$

➜ Gap caused by pre-training and not architecture

Token-Independent Text Encoder (TITE)

Conclusion

TITE outputs a single sequence-level vector for an input sequence.

Token-Independent Text Encoder (TITE)

Conclusion

TITE outputs a single sequence-level vector for an input sequence.

- Around 2× faster than flash BERT for encoding a sequence

Token-Independent Text Encoder (TITE)

Conclusion

TITE outputs a single sequence-level vector for an input sequence.

- ❑ Around 2× faster than flash BERT for encoding a sequence
- ❑ Same effectiveness as standard bi-encoder models

Token-Independent Text Encoder (TITE)

Conclusion

TITE outputs a single sequence-level vector for an input sequence.

- Around 2× faster than flash BERT for encoding a sequence
- Same effectiveness as standard bi-encoder models
- **BUT:** Further work is necessary to be on par with SOTA models

Token-Independent Text Encoder (TITE)

Conclusion

TITE outputs a single sequence-level vector for an input sequence.

- Around 2× faster than flash BERT for encoding a sequence
- Same effectiveness as standard bi-encoder models
- **BUT:** Further work is necessary to be on par with SOTA models

Paper will be presented at SIGIR 2025. Pre-print available, models coming soon.

Code and paper @
webis-de/tite

Token-Independent Text Encoder (TITE)

Conclusion

TITE outputs a single sequence-level vector for an input sequence.

- Around 2× faster than flash BERT for encoding a sequence
- Same effectiveness as standard bi-encoder models
- **BUT:** Further work is necessary to be on par with SOTA models

Paper will be presented at SIGIR 2025. Pre-print available, models coming soon.

Code and paper @
webis-de/tite

Coding Tutorial
13.06.2024 at 2 PM

Code and paper @
webis-de/set-encoder

Token-Independent Text Encoder (TITE)

Conclusion

TITE outputs a single sequence-level vector for an input sequence.

- Around 2× faster than flash BERT for encoding a sequence
- Same effectiveness as standard bi-encoder models
- **BUT:** Further work is necessary to be on par with SOTA models

Paper will be presented at SIGIR 2025. Pre-print available, models coming soon.

Code and paper @
webis-de/tite

Coding Tutorial
13.06.2024 at 2 PM

Code and paper @
webis-de/set-encoder

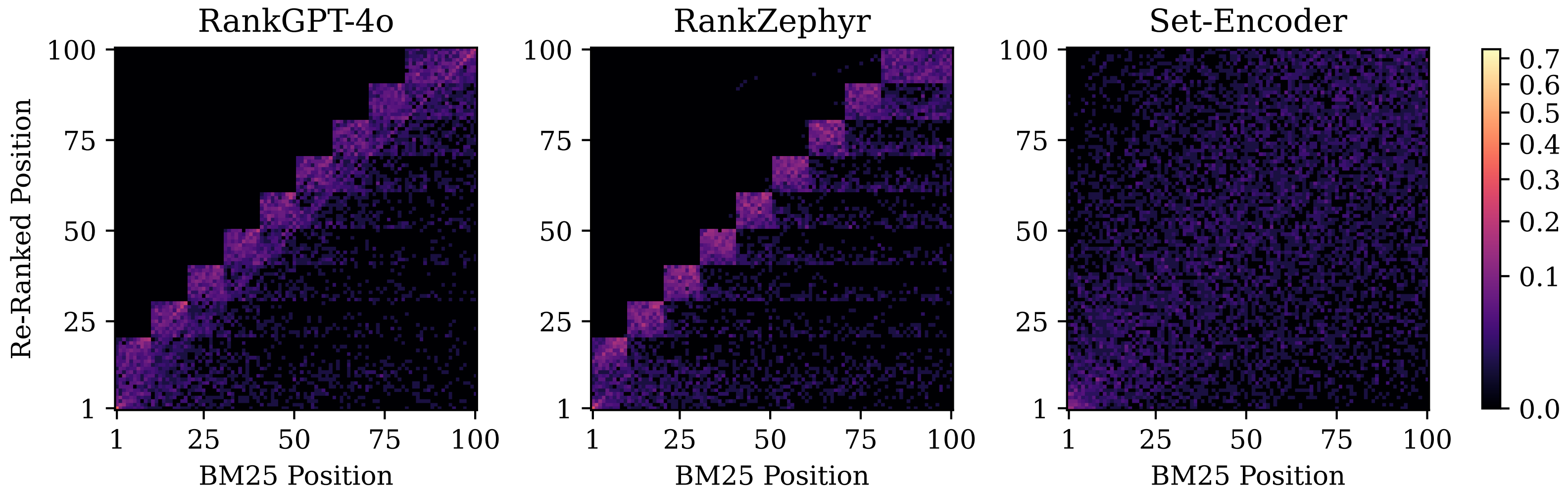
Thanks!

Set-Encoder

Rank Changes

Previous listwise re-rankers are biased by the order of the input documents.

What do these biases look like? ➜ Plot the original vs re-ranked position.



A substantial number of previous works attempt to mitigate these positional biases.
[Zhuang et al., SIGIR’24; Parry et al., arXiv’24]

➜ Making the model permutation-invariant is a more principled approach.

Set-Encoder

TREC DL

Model	Size	TREC DL 19		TREC DL 20	
		BM25	CBv2	BM25	CBv2
First Stage	–	0.480†	0.732†	0.494†	0.724†
RankGPT-4o	N/A	0.725	0.784	0.719	0.793
RankGPT-4o Full	N/A	0.732	0.781	0.711	0.796
RankZephyr	7B	0.719	0.749	0.720	0.798
LiT5-Distill	220M	0.696	0.753	0.679†	0.744†
monoT5 3B	3B	0.705	0.745	0.715	0.757
RankT5 3B	3B	0.710	0.752	0.711	0.772
monoELECTRA	110M	0.720	0.768	0.711†	0.770
	330M	**0.733**	0.765	0.727	**0.799**
Set-Encoder	110M	0.724	0.788	0.710†	0.777
	330M	0.727	**0.789**	**0.735**	0.790

Set-Encoder

TREC DL Novelty

	Model	Prompt / Loss		nDCG		α-nDCG	
				2019	2020	2019	2020
(1)	First Stage	–		0.732	0.724	$0.700^{\dagger}$	$0.722^{\dagger}$
(2)	RankGPT-4o	Relevance		$0.784^{\dagger}$	$0.793^{\dagger}$	0.750	0.759
(3)		Novelty		$0.778^{\dagger}$	**0.806**†	0.741	0.773
(4)	RankGPT-4o Full	Relevance		$0.781^{\dagger}$	$0.796^{\dagger}$	0.738	0.763
(5)		Novelty		$0.785^{\dagger}$	$0.803^{\dagger}$	0.750	0.771
(6)	RankZephyr	Relevance		0.749	$0.798^{\dagger}$	$0.699^{\dagger}$	0.765
(7)		Novelty		0.753	$0.800^{\dagger}$	$0.700^{\dagger}$	0.760
(8)	**Model**	**1st $\mathcal{L}$**	**2nd $\mathcal{L}$**				
(9)	monoELECTRA	$\mathcal{L}_{\text{InfoNCE}}$	$\mathcal{L}_{\text{RankNet}}$	$0.768^{\dagger}$	$0.770^{\dagger}$	$0.718^{\dagger}$	$0.745^{\dagger}$
(10)			$\mathcal{L}_{\text{NA-RankNet}}$	0.704	0.675	0.785	0.753
(11)	Set-Encoder	$\mathcal{L}_{\text{InfoNCE}}$	$\mathcal{L}_{\text{RankNet}}$	$0.780^{\dagger}$	$0.757^{\dagger}$	$0.733^{\dagger}$	$0.747^{\dagger}$
(12)			$\mathcal{L}_{\text{NA-RankNet}}$	0.714	0.651	0.779	$0.743^{\dagger}$
(13)		$\mathcal{L}_{\text{DA-InfoNCE}}$	$\mathcal{L}_{\text{RankNet}}$	**0.788**†	$0.777^{\dagger}$	$0.740^{\dagger}$	$0.752^{\dagger}$
(14)			$\mathcal{L}_{\text{NA-RankNet}}$	0.710	0.690	**0.821**	**0.803**
(15)	Set-Enc. ~~[INT]~~	$\mathcal{L}_{\text{DA-InfoNCE}}$	$\mathcal{L}_{\text{NA-RankNet}}$	0.707	0.670	0.773	$0.748^{\dagger}$

Set-Encoder

Out-of-Domain Re-Ranking

Model	**Size**	**Antique**	**Args.me**	**ClueWeb09**	**ClueWeb12**	**CORD-19**	**Cranfield**	**Disks4+5**	**GOV**	**GOV2**	**MEDLINE**	**NFCorpus**	**Vaswani**	**WaPo**	**G. Mean**
First Stage	–	.516†	.405	.177†	**.364**†	.586†	**.012**	.424†	.259†	.467†	.385	.281†	.447†	.364†	.286
RankZephyr	7B	.534†	.364†	.213	.303	**.767**†	.009	.542	.349	.560	**.460**†	.314	.512	**.508**	.320
LiT5-Distill	220M	.576†	.395	.214	.275†	.686	.011	.495†	.304†	.534†	.354†	.293†	.429†	.470	.302
monoT5 3B	3B	.590	.415	.188†	.323	.649†	.011	.526	.345	.529†	.395	.319	.474†	.469	.313
RankT5 3B	3B	.598	**.421**	**.227**	.336	.713	.010	.538	**.353**	.528†	.406	**.323**	.459†	.468†	**.322**
m.ELECTRA	110M	.593	.375†	.209	.295	.692	.010	.507†	.305†	.541†	.399	.306	.522	.458†	.309
	330M	.575†	.369†	.221	.313	.716	.008	**.546**	.344	.572	.419	.316	.526	.504	.318
Set-Encoder	110M	.594	.375†	.216	.299	.683	.010	.513†	.306†	.543†	.396	.306	.523	.461†	.311
	330M	**.606**	.409	.226	.310	.702	.009	.534	.334	**.573**	.405	.313	**.530**	**.508**	.321

Set-Encoder

Efficiency

Model	**Size**	**Time**	**Memory**
RankGPT-4o	N/A	18.773	N/A
RankGPT-4o Full	N/A	7.362	N/A
RankZephyr	7B	24.047	15.48
LiT5-Distill	220M	2.054	2.69
monoT5$_{3B}$	3B	0.998	29.36
RankT5$_{3B}$	3B	0.942	29.04
monoELECTRA	110M	0.139	1.18
	330M	0.215	2.69
Set-Encoder	110M	0.147	1.25
	330M	0.219	2.60

TITE

Efficiency

Model						**Kernel**	**Queries**	**Documents**
BERT						Eager	24.0 (0.5×)	2.0 (0.2×)
DistilBERT						Eager	47.1 (1.0×)	3.7 (0.4×)
Funnel Transformer						Eager	14.2 (0.3×)	1.3 (0.1×)
TITE (Pool Param.: ①)						Eager	69.8 (1.5×)	6.7 (0.8×)
BERT						SDPA	28.9 (0.6×)	3.2 (0.4×)
DistilBERT						SDPA	57.9 (1.2×)	6.4 (0.7×)
TITE (Pool Param.: ①)						SDPA	81.2 (1.7×)	13.4 (1.5×)
BERT						Flash	48.0	8.7
ModernBERT						Flash	41.1 (0.9×)	8.3 (1.0×)
		k,s	Arr.	Loc.	Dim.			
TITE	①	2	L	Intra	768	Flash	89.0 (1.9×)	20.8 (2.4×)
	②	2	S	Intra	768	Flash	96.0 (2.0×)	28.5 (3.3×)
	③	3	L	Intra	768	Flash	68.1 (1.4×)	14.5 (1.7×)
	④	3	S	Intra	768	Flash	94.6 (2.0×)	30.8 (3.5×)
	⑤	2	L	Pre	768	Flash	89.6 (1.9×)	21.2 (2.4×)
	⑥	2	L	Post	768	Flash	89.0 (1.9×)	20.3 (2.3×)
	⑦	2	L	Intra	1536	Flash	70.1 (1.5×)	16.3 (1.9×)

TITE

Effectiveness

Model	TREC DL		BEIR															
	2019	2020	ArguAna	Climate-FEVER	CQADupStack	DBPedia	FEVER	FiQA	HotpotQA	NFCorpus	NQ	Quora	SCIDOCS	SciFact	TREC-COVID	Touché	Arith. Mean	Geom. Mean
BM25	.506†	.480†	.397†	.165†	.302	.318†	.651†	.236†	.633†	.322	.305†	.789†	.149	.679†	.595†	**.442**†	.427	.379
S-BERT (Repro.)	.700	.688	.336	.224	.319	.369	.727	.317	.574	.303	.510	.844	.146	.603	.756	.256	.449	.399
S-BERT	.705	.726	.384†	.221	.337	.385	.762†	.323	.585†	.315	.522†	.844	.146	.606	.744	.237	.458	.407
S-DistilBERT	.705	.699	.355†	.233	.322	.375	.774†	.286†	.571	.298	.497†	.833†	.140	.596	.666†	.224	.441	.391
S-ModernBERT	–	–	.357	.236	.331	.238	.599	.288	.461	.237	.395	.859	.125	.570	.721	.208	.402	.352
RetroMAE (Repro.)	.723	.711	.375†	**.242**†	.340	.406†	.737†	.340†	.624†	.336†	.539†	.844	**.163**†	.663†	**.780**	.273	.476	.428
RetroMAE	.712	**.730**	.367†	.240†	.342	.428†	.777†	.343†	.668†	.325†	**.573**†	**.853**†	.160†	.638	.759	.280	.482	.432
ColBERTv2	**.732**	.724	.453†	.176†	**.359**	**.441**†	.774†	.346†	.665†	.330†	.547†	.851†	.150	.691†	.732	.257	.484	.427
SPLADE++	.731	.720	**.520**†	.230	.334	.437†	**.788**†	**.347**†	**.687**†	**.347**†	.538†	.834†	.159†	**.704**†	.727	.247	**.493**	**.440**
TITE (Base)	.705	.670	.391†	.204†	.312	.376	.699†	.302	.604†	.334†	.484†	.818†	.156	.647†	.691	.271	.449	.403
TITE (Upscale)	.724	.686	.373†	.209†	.323	.374	.704†	.298	.616†	.328†	.490†	.827†	.155	.632	.715	.275	.451	.404

Model Parameters						TREC DL		BEIR	
		k,s	Arr.	Loc.	Dim.	2019	2020	Arith.	Geom.
TITE	①	2	L	Intra	768	.705	.670	.449	.403
	②	2	S	Intra	768	.675	.663	.443	.397
	③	3	L	Intra	768	.683	.672	.445	.400
	④	3	S	Intra	768	.673	.669	.443	.399
	⑤	2	L	Pre	768	.686	.682	.445	.400
	⑥	2	L	Post	768	.670	.683	.446	.399
	⑦	2	L	Intra	1536	.724	.686	.451	.404