

# Fast Passage Re-ranking with Contextualized Exact Term Matching and Efficient Passage Expansion

Shengyao Zhuang & Guido Zuccon

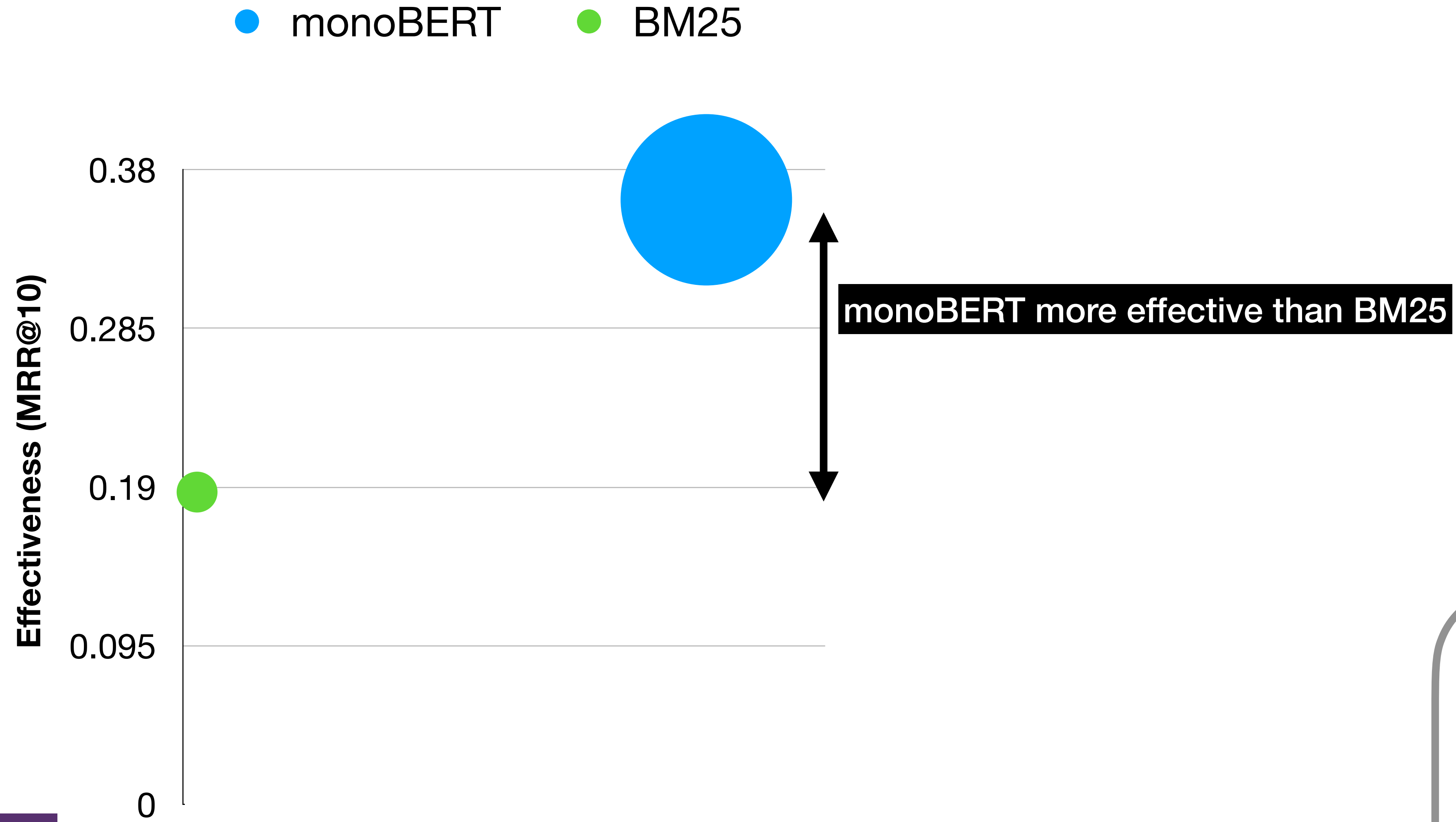
{s.zhuang,g.zuccon}@uq.edu.au

ielab, The University of Queensland, Australia

[www.ielab.io](http://www.ielab.io)

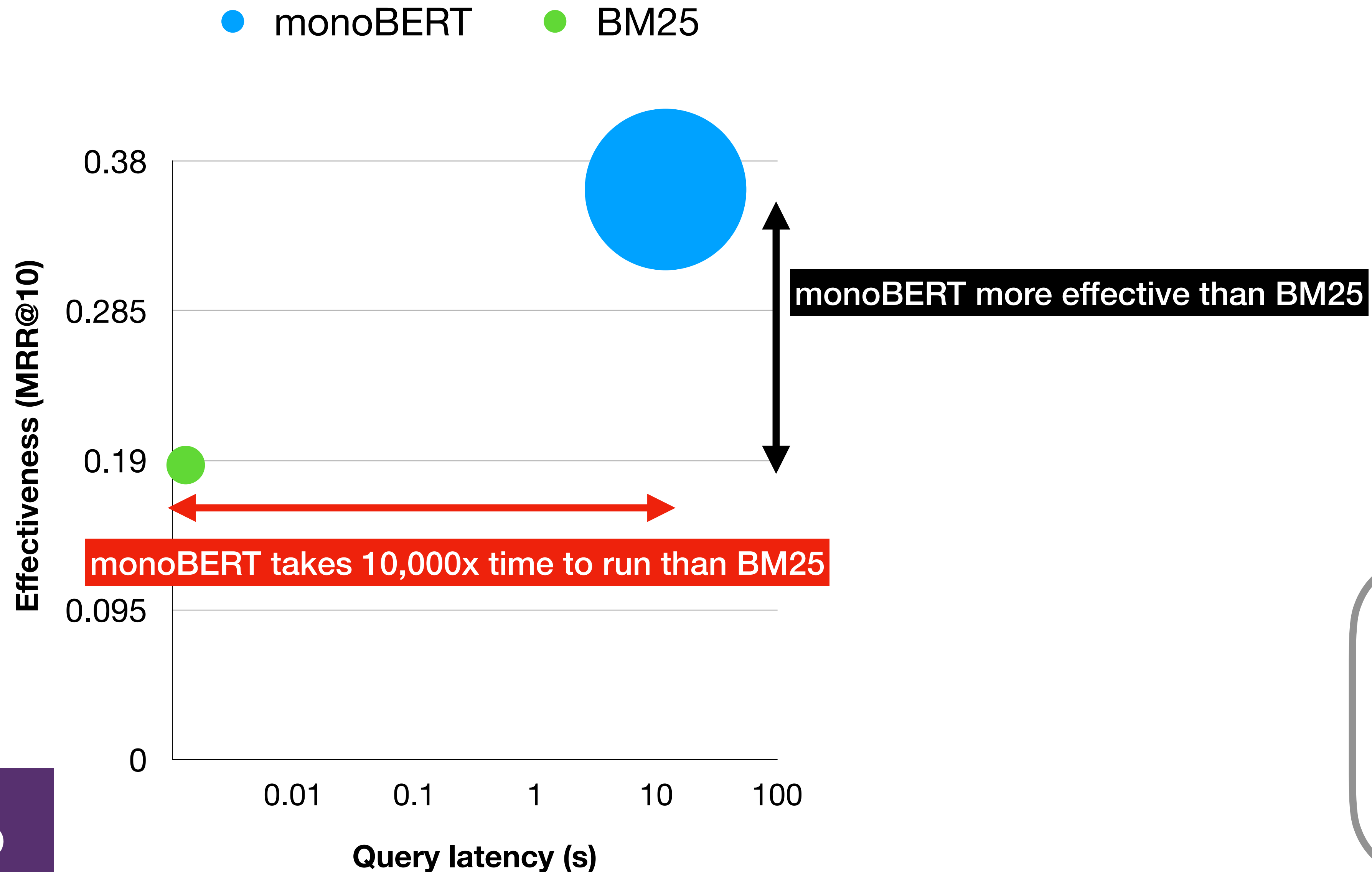


# monoBERT is effective!



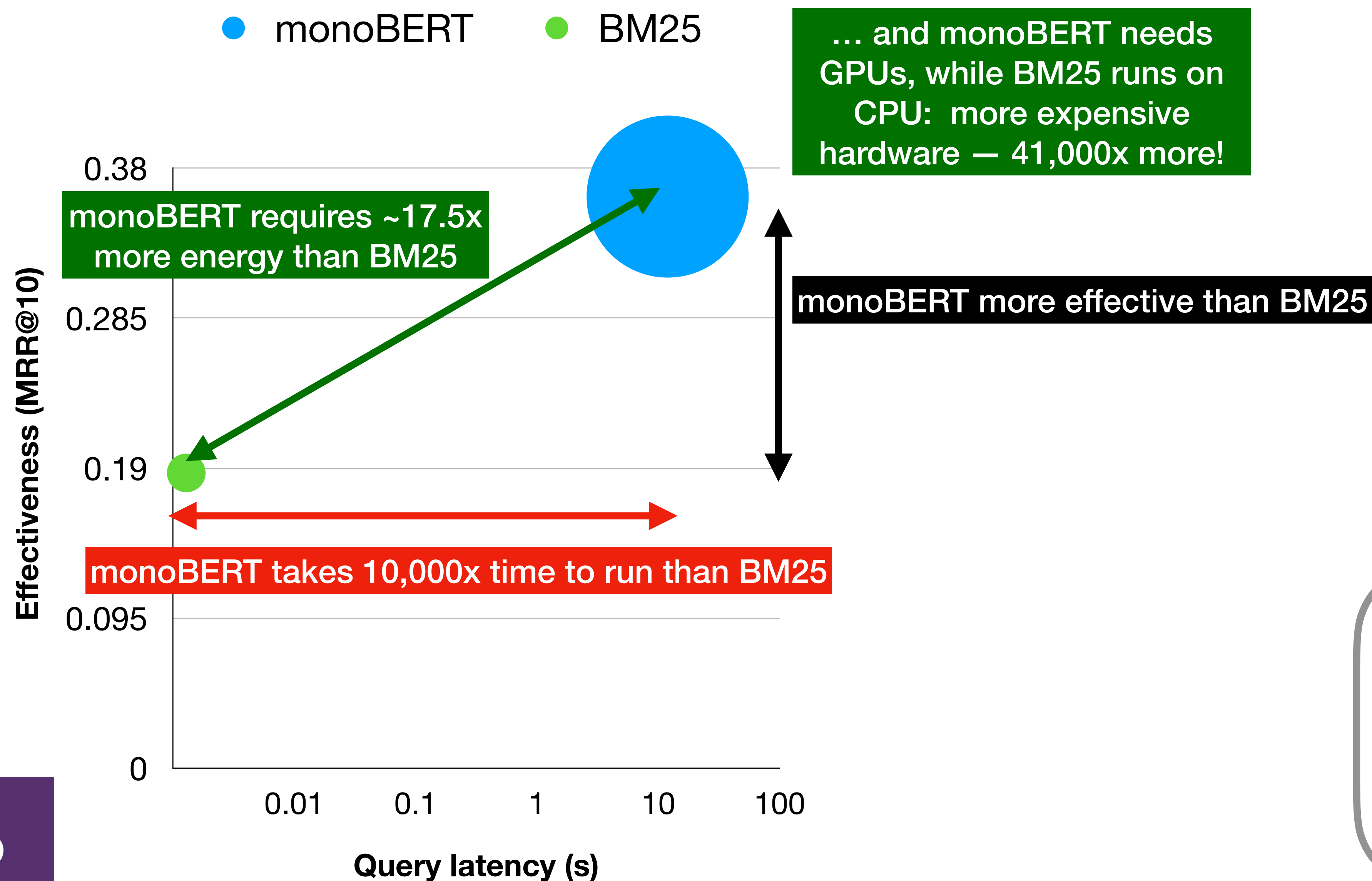
Scells, Zhuang, Zuccon,  
“Reduce, Reuse, Recycle: Green  
Information Retrieval Research”,  
SIGIR 2022

# monoBERT's Challenges: (1) it's slow



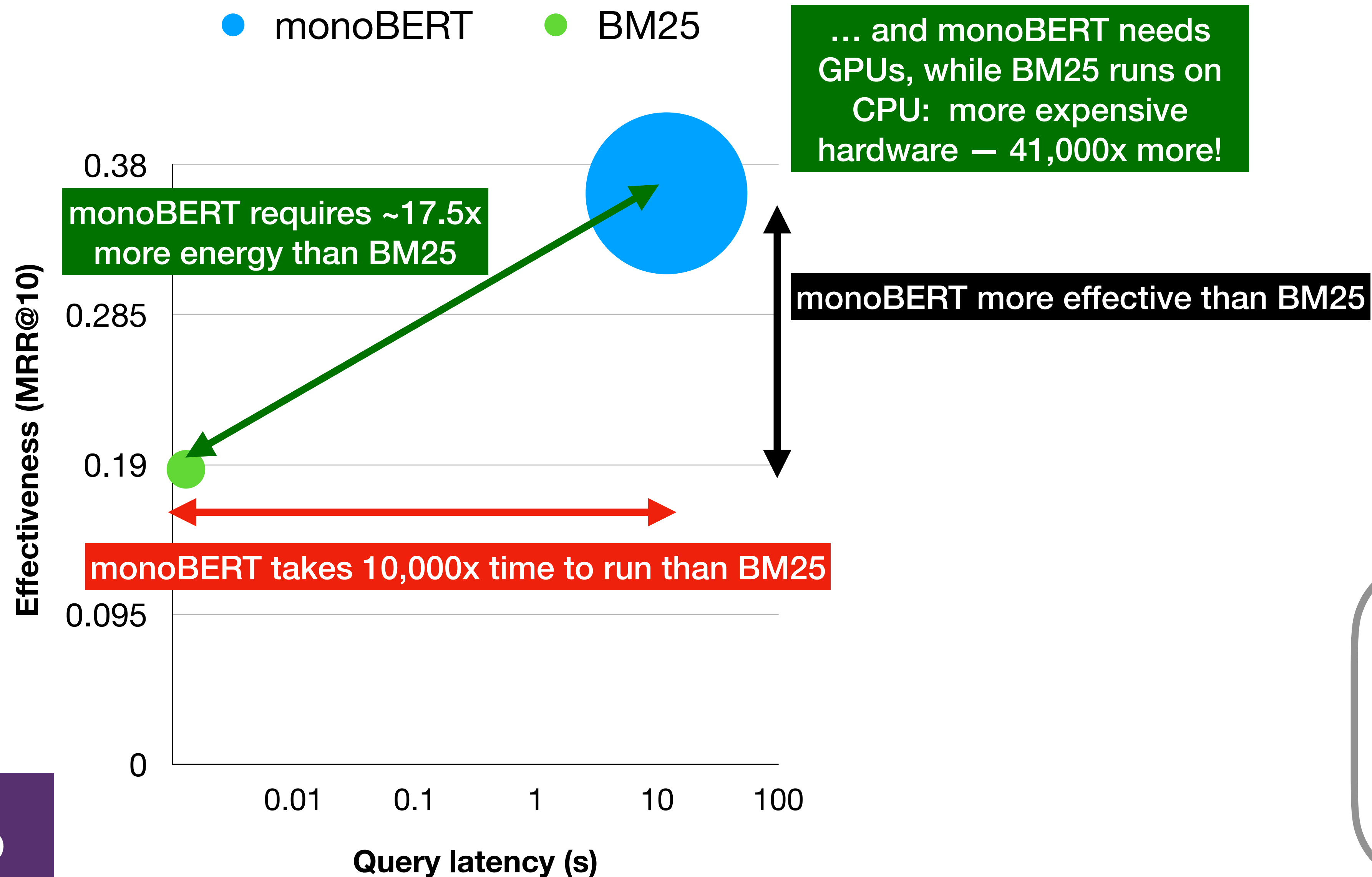
Scells, Zhuang, Zuccon,  
"Reduce, Reuse, Recycle: Green  
Information Retrieval Research",  
SIGIR 2022

# monoBERT's Challenges: (1) it's slow, (2) it's expensive



Scells, Zhuang, Zuccon,  
"Reduce, Reuse, Recycle: Green  
Information Retrieval Research",  
SIGIR 2022

# BERT's Challenges: (1) it's slow, (2) it's expensive



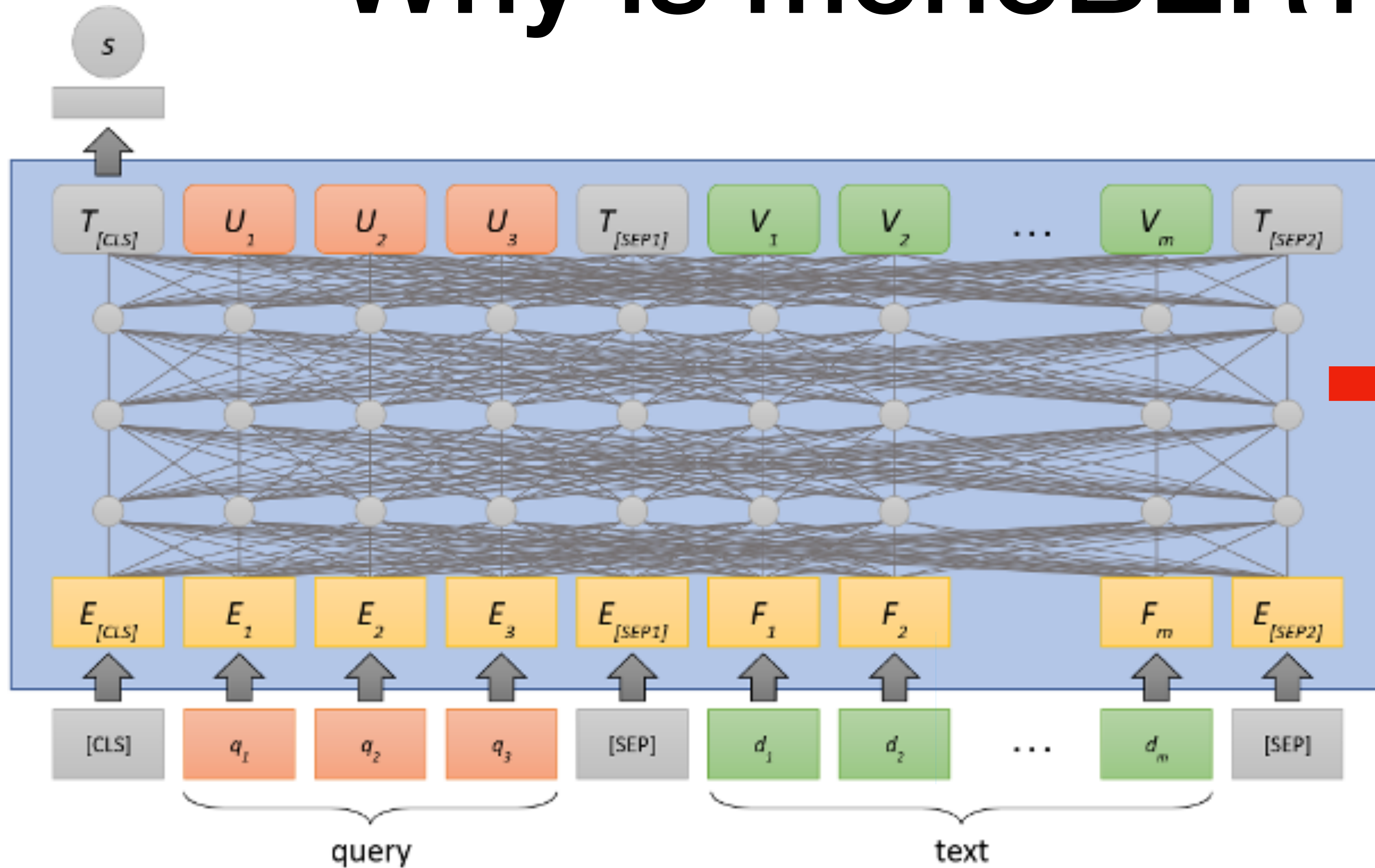
monoBERT produces 165,000x more CO<sub>2</sub> emissions than BM25



Scells, Zhuang, Zuccon,  
“Reduce, Reuse, Recycle: Green Information Retrieval Research”,  
SIGIR 2022



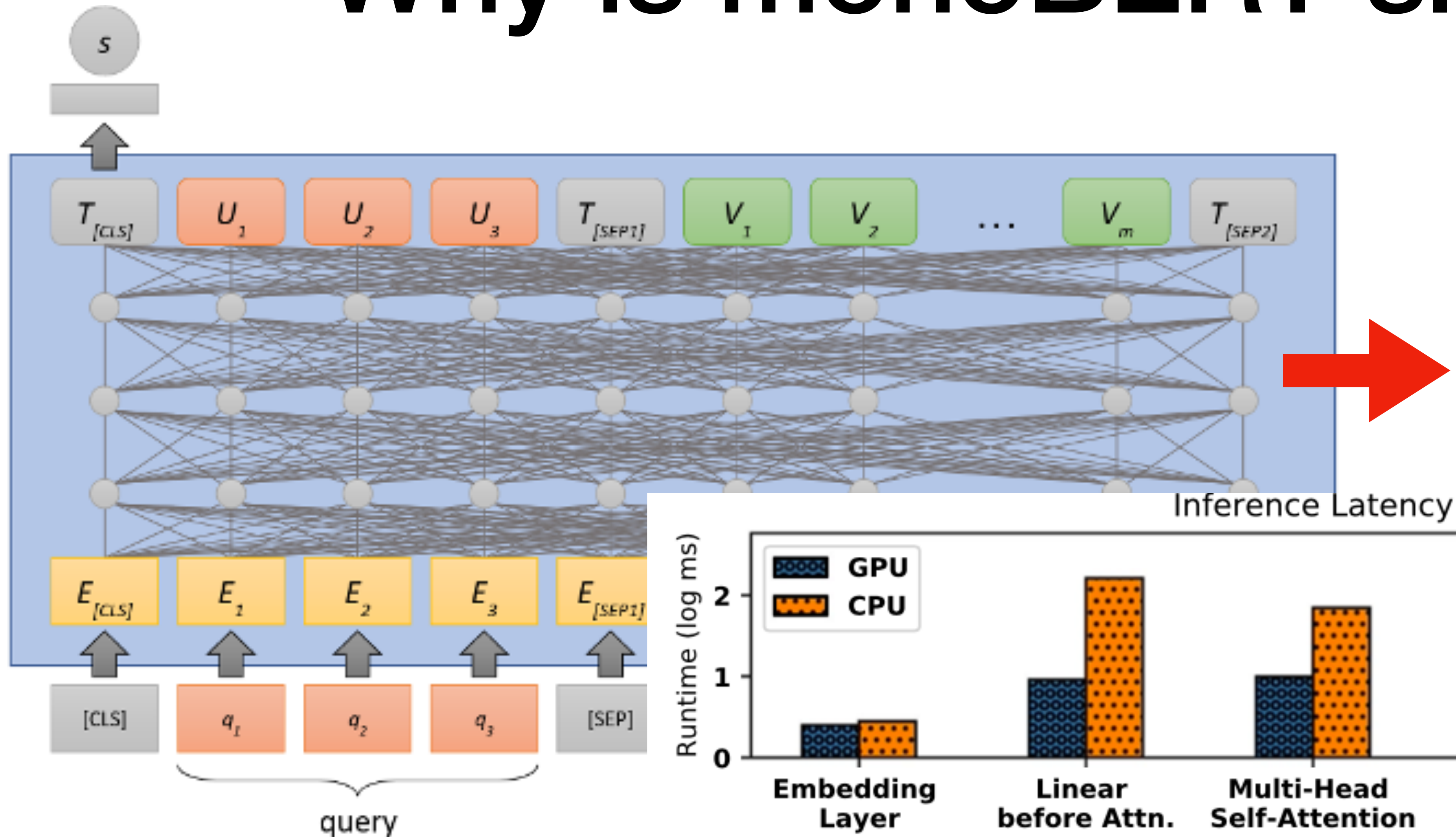
# Why is monoBERT slow?



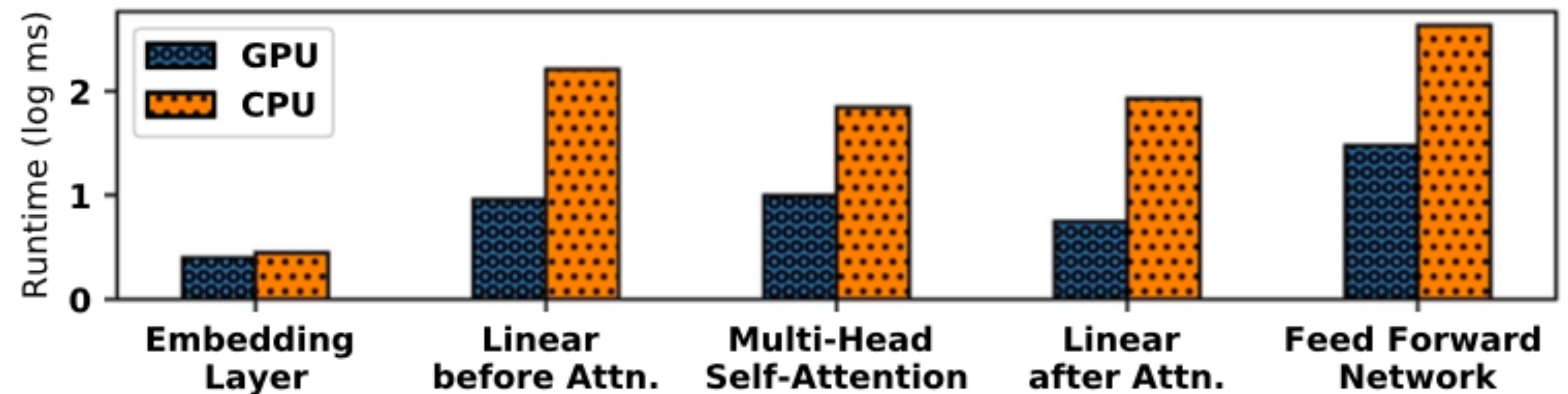
**Computationally expensive layers**  
e.g. 110+ million learned weights



# Why is monoBERT slow?



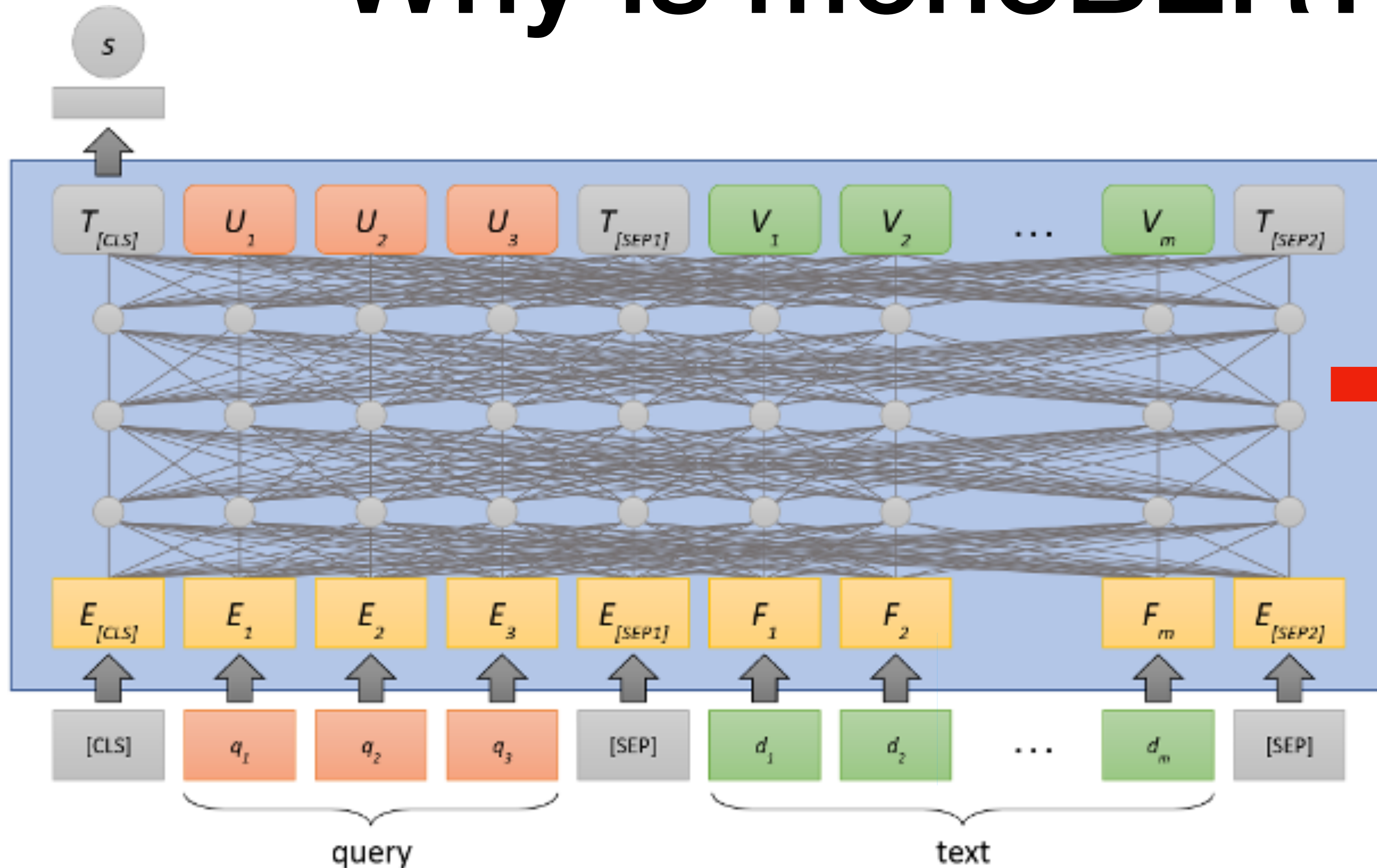
**Computationally expensive layers**  
e.g. 110+ million learned weights



Ganesh et al. "Compressing Large-Scale Transformer-Based Models: A Case Study on BERT", Trans. ACL, 2021



# Why is monoBERT slow?

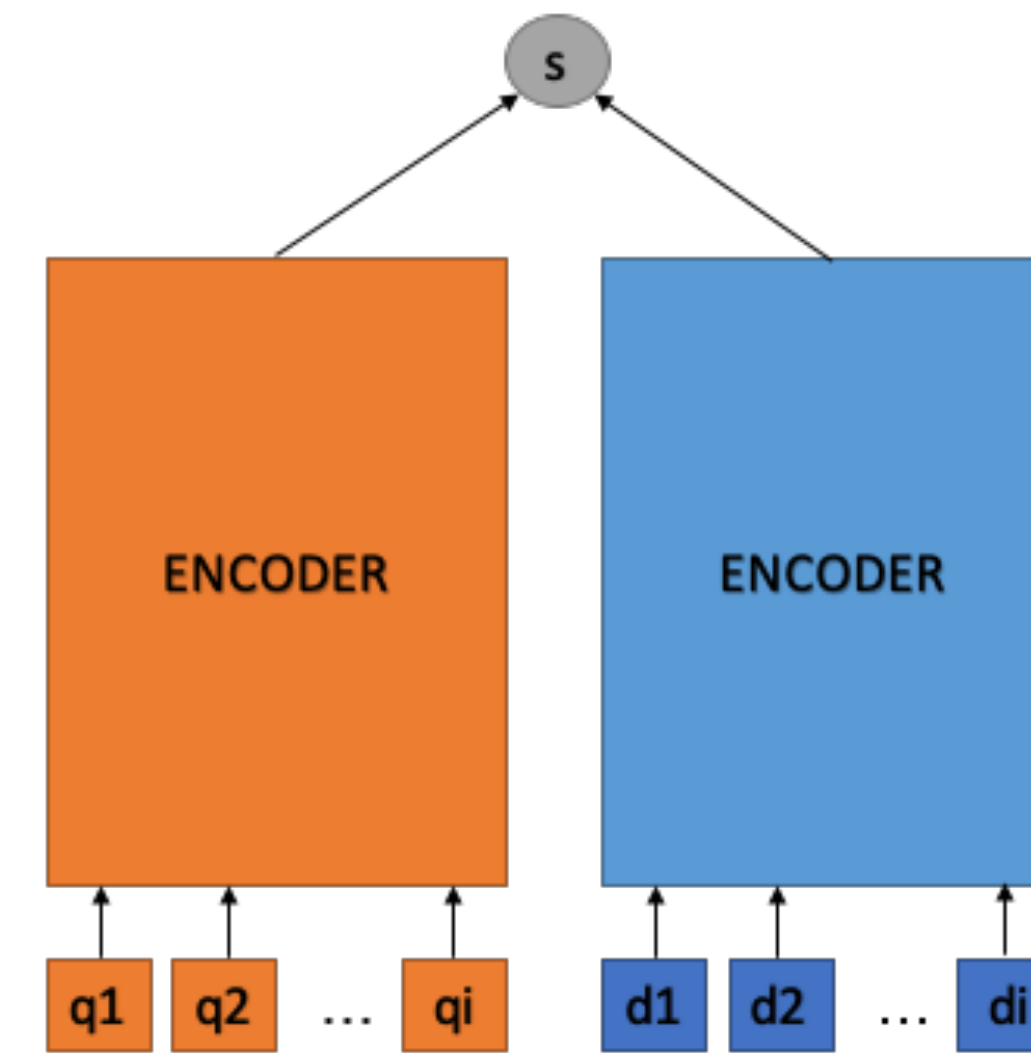
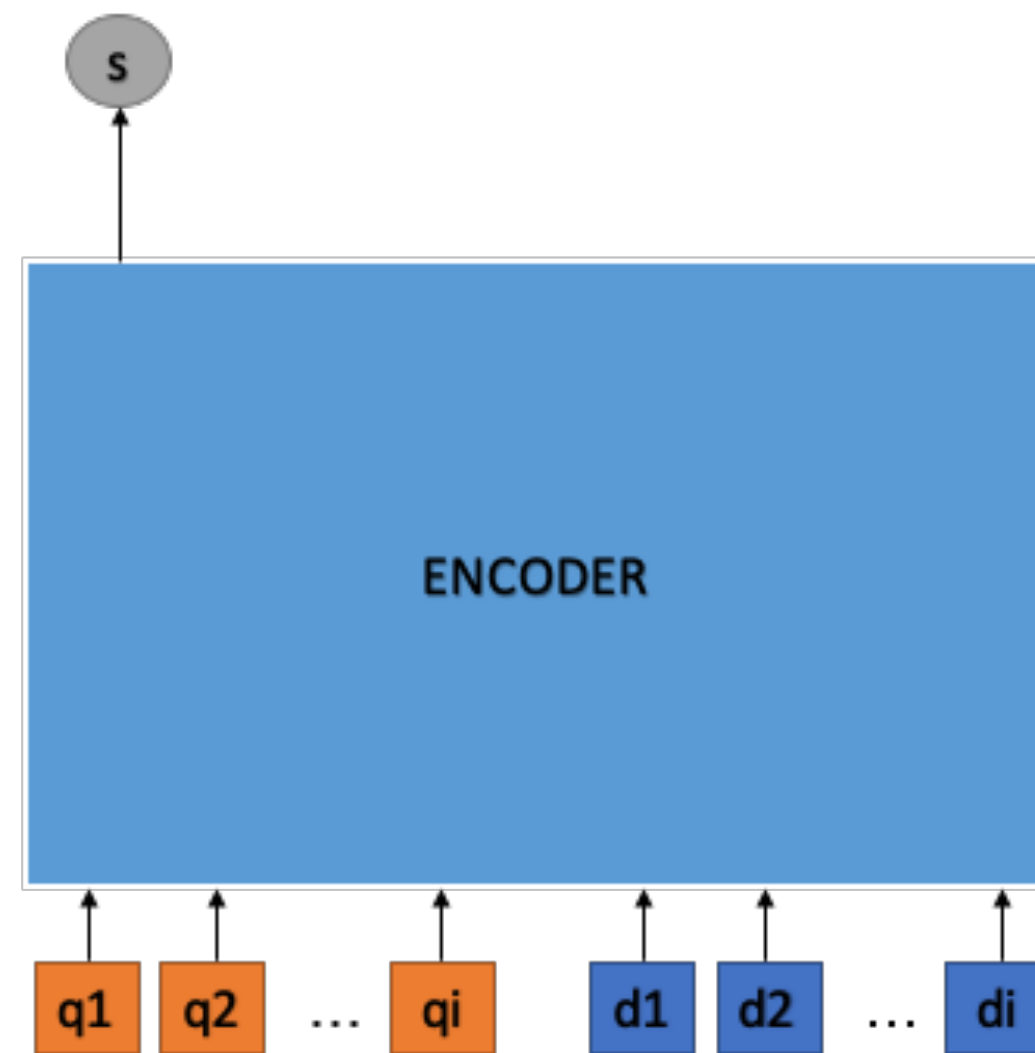


**Computationally expensive layers**  
e.g. 110+ million learned weights

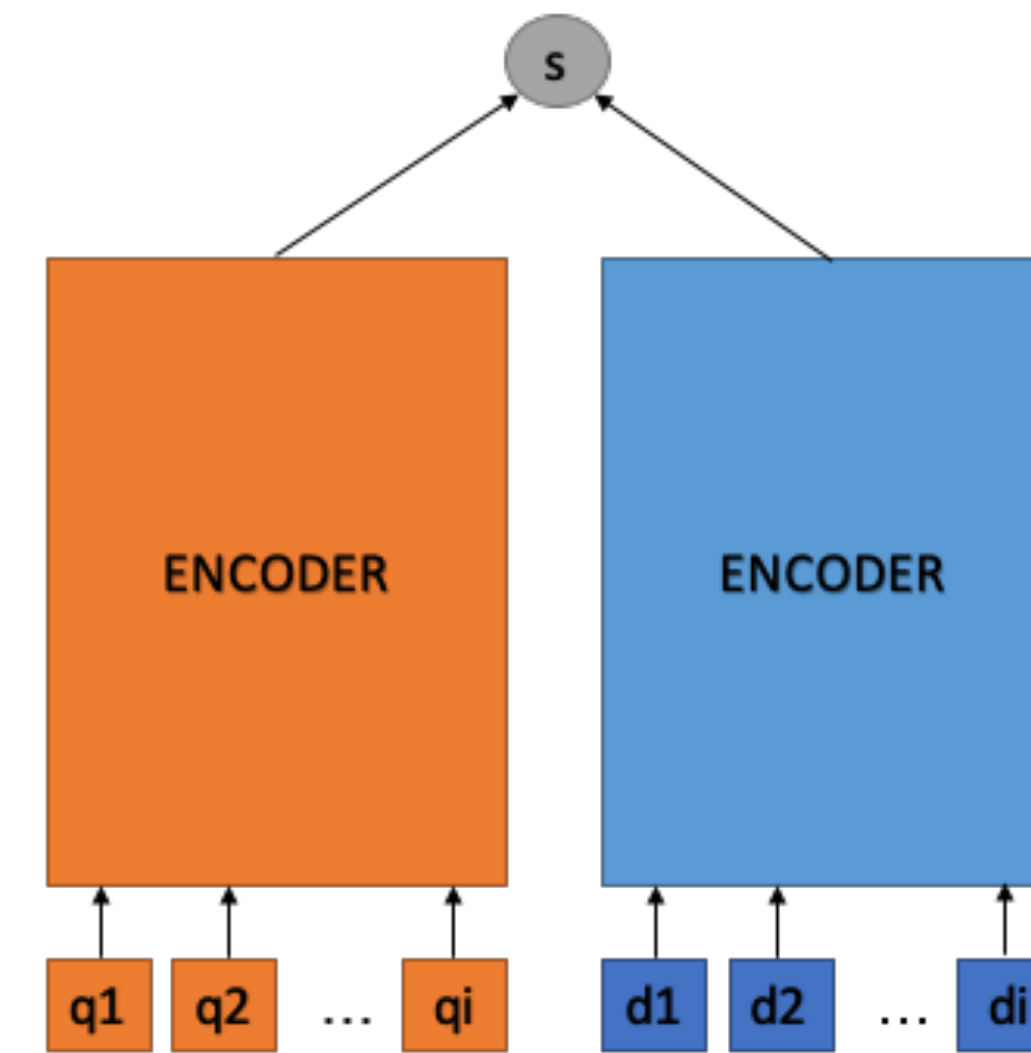
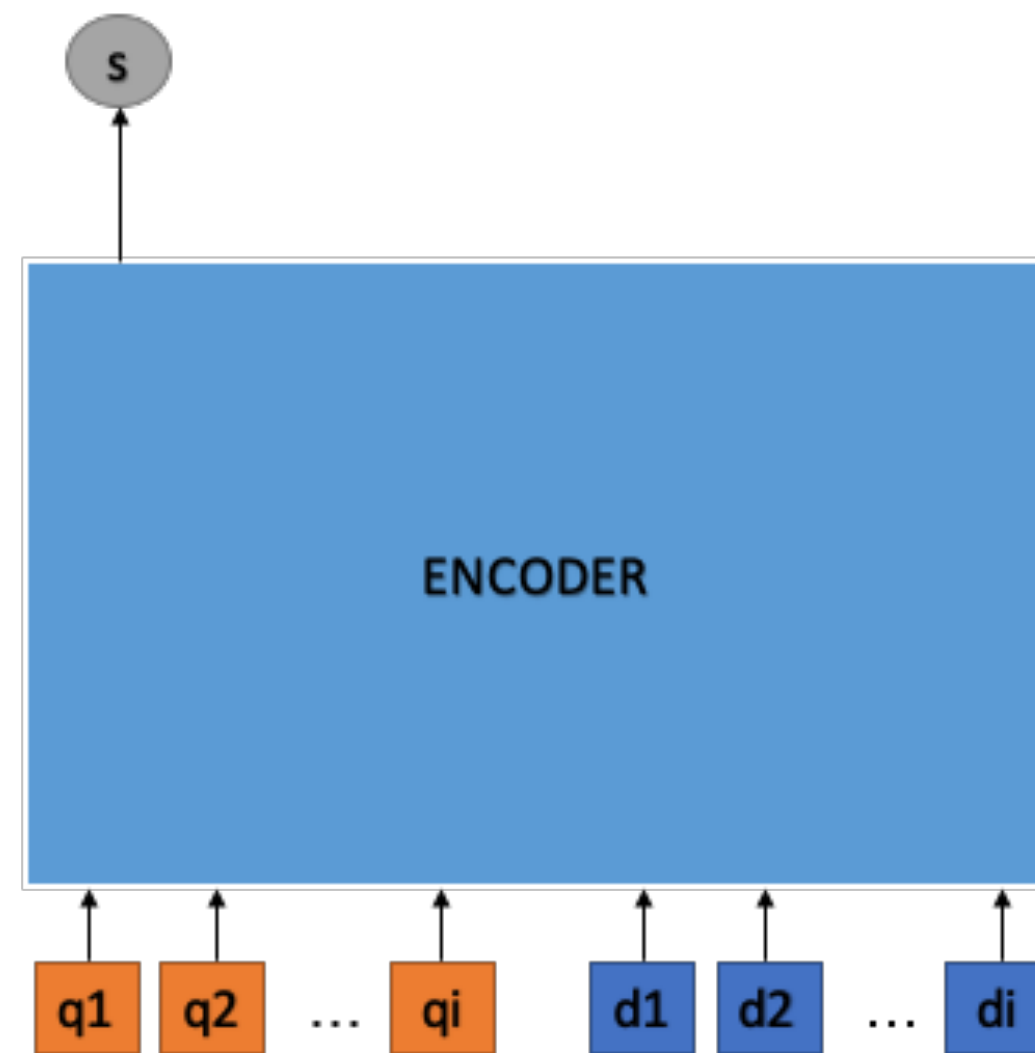
Possible Solutions:  
A. Multistage ranking pipeline with limited re-ranking  
B. Simplification of BERT inference to lower query latency



# Cross-Encoder, Bi-encoder, ... can we simplify further?



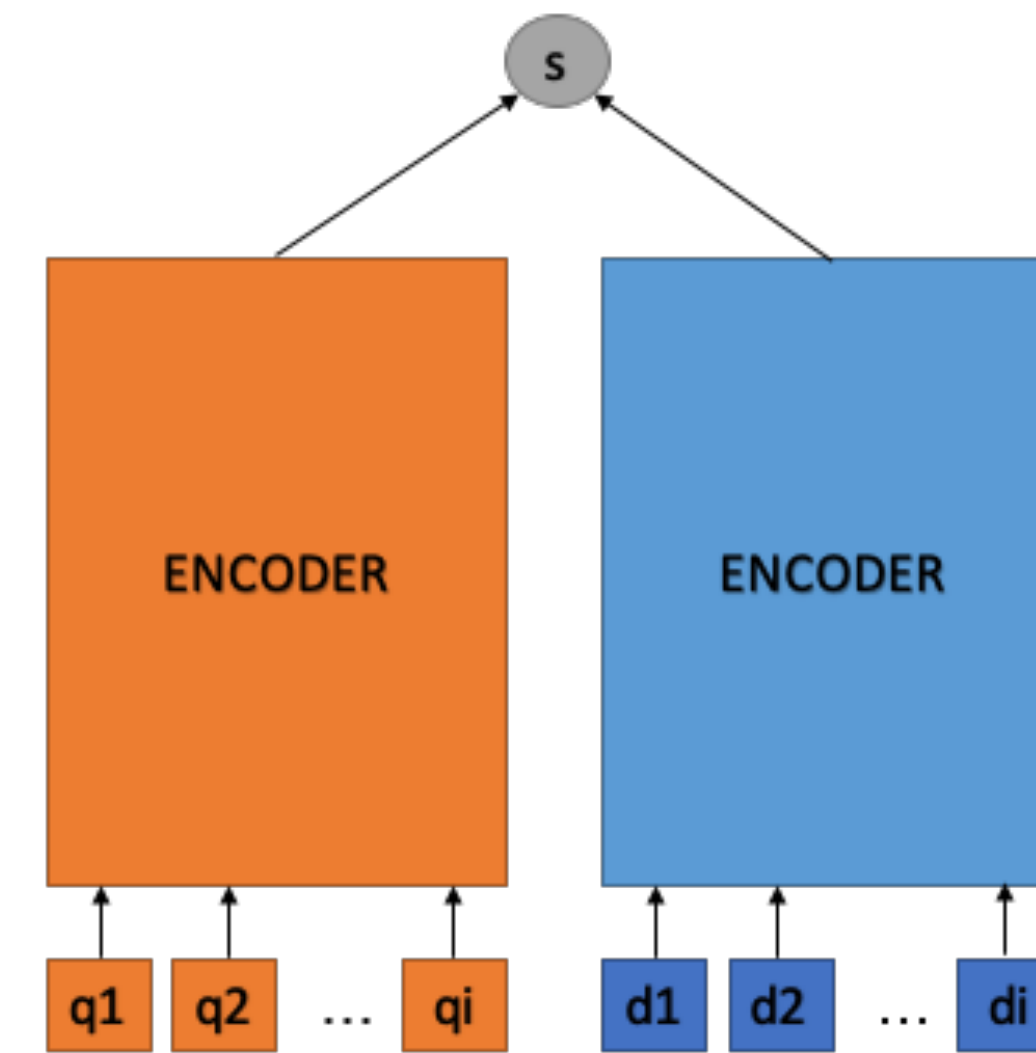
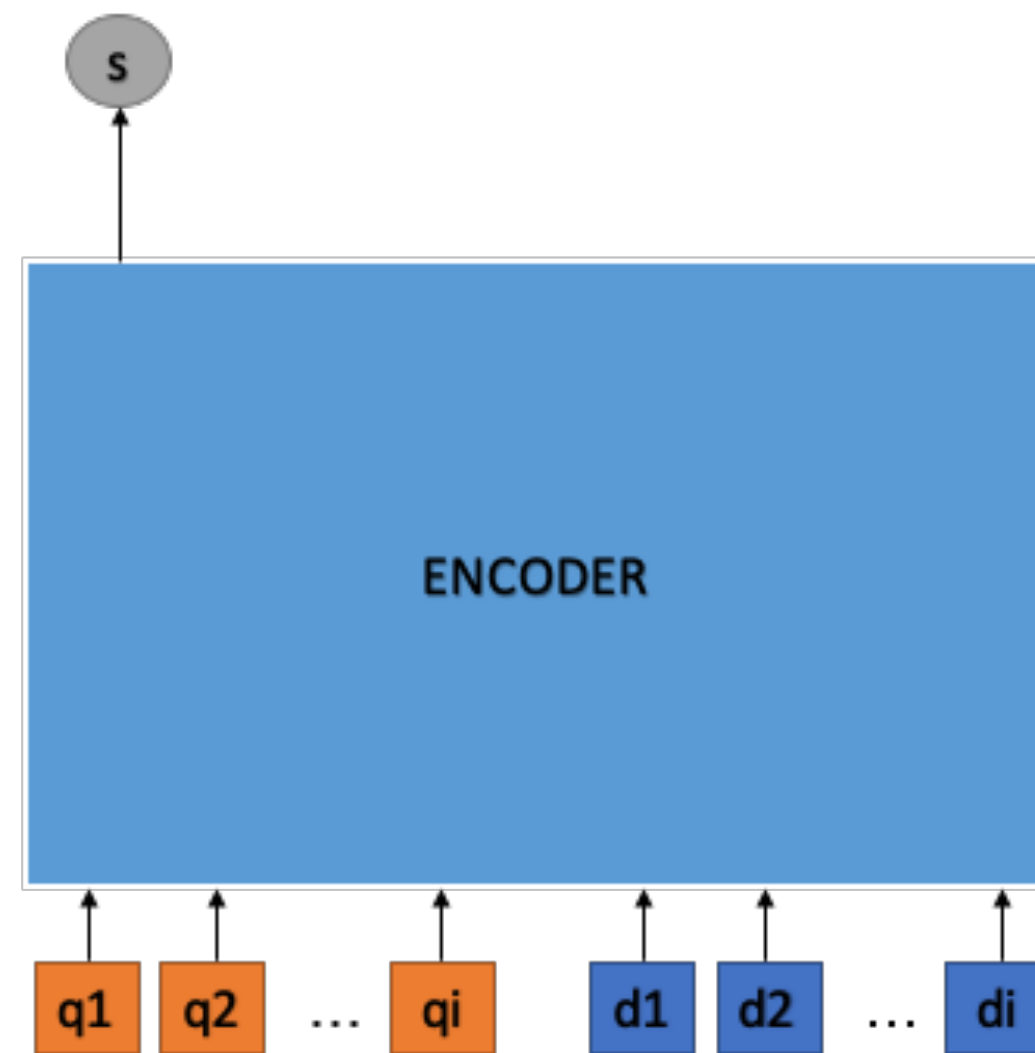
# Cross-Encoder, Bi-encoder, ... can we simplify further?



If wanting to reduce query latency,  
then no need to modify this

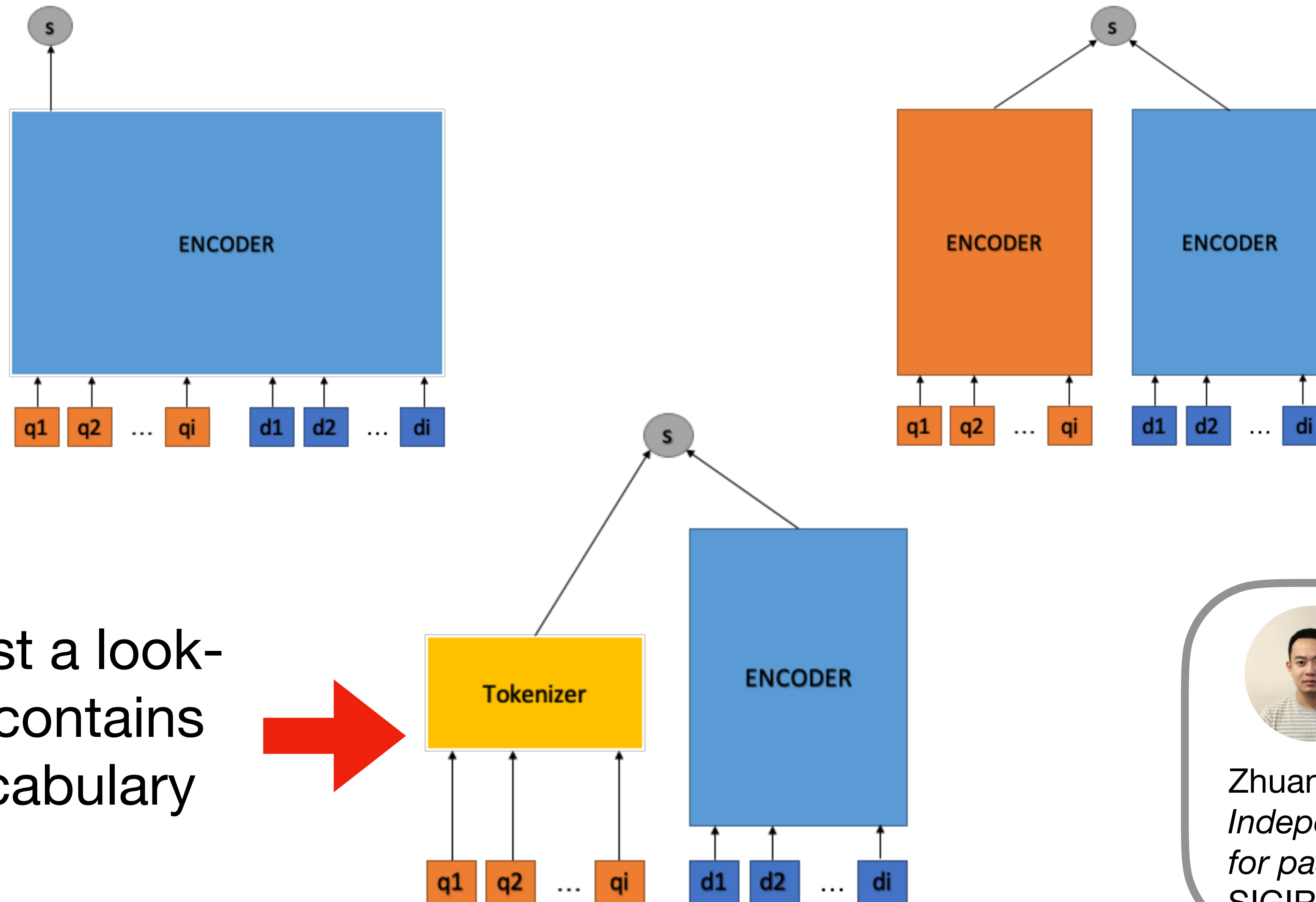


# Cross-Encoder, Bi-encoder, ... can we simplify further?

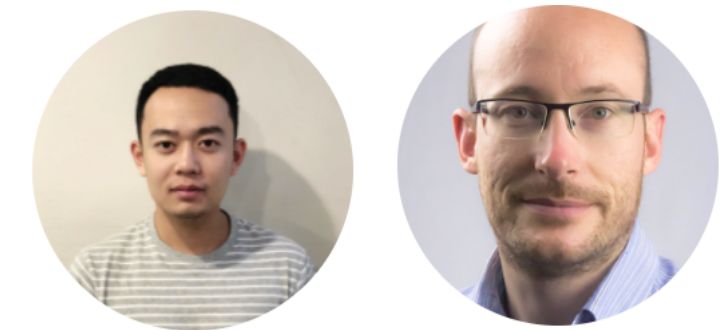


This influences query latency — can we make this faster?

# Cross-Encoder, Bi-encoder, ... can we simplify further?



Now, this is just a look-up table that contains the BERT vocabulary



Zhuang, Zuccon, "TILDE: Term Independent Likelihood model for passage re-ranking", SIGIR 2021



# TILDE: Term Independent Likelihood moDEI

- Use BERT tokeniser to obtain sparse query encoding
- Use CLS token to encode document
- Project CLS token embedding to  $|V|$  vector
- Inner product between sparse query vector and document vector, in the  $|V|$  space

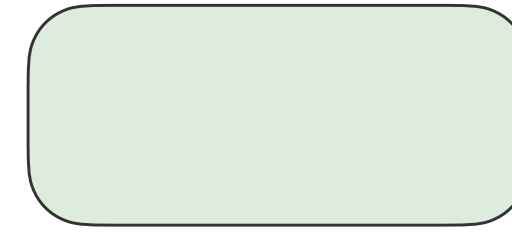


Zhuang, Zuccon, “*TILDE: Term Independent Likelihood moDEI for passage re-ranking*”,  
SIGIR 2021

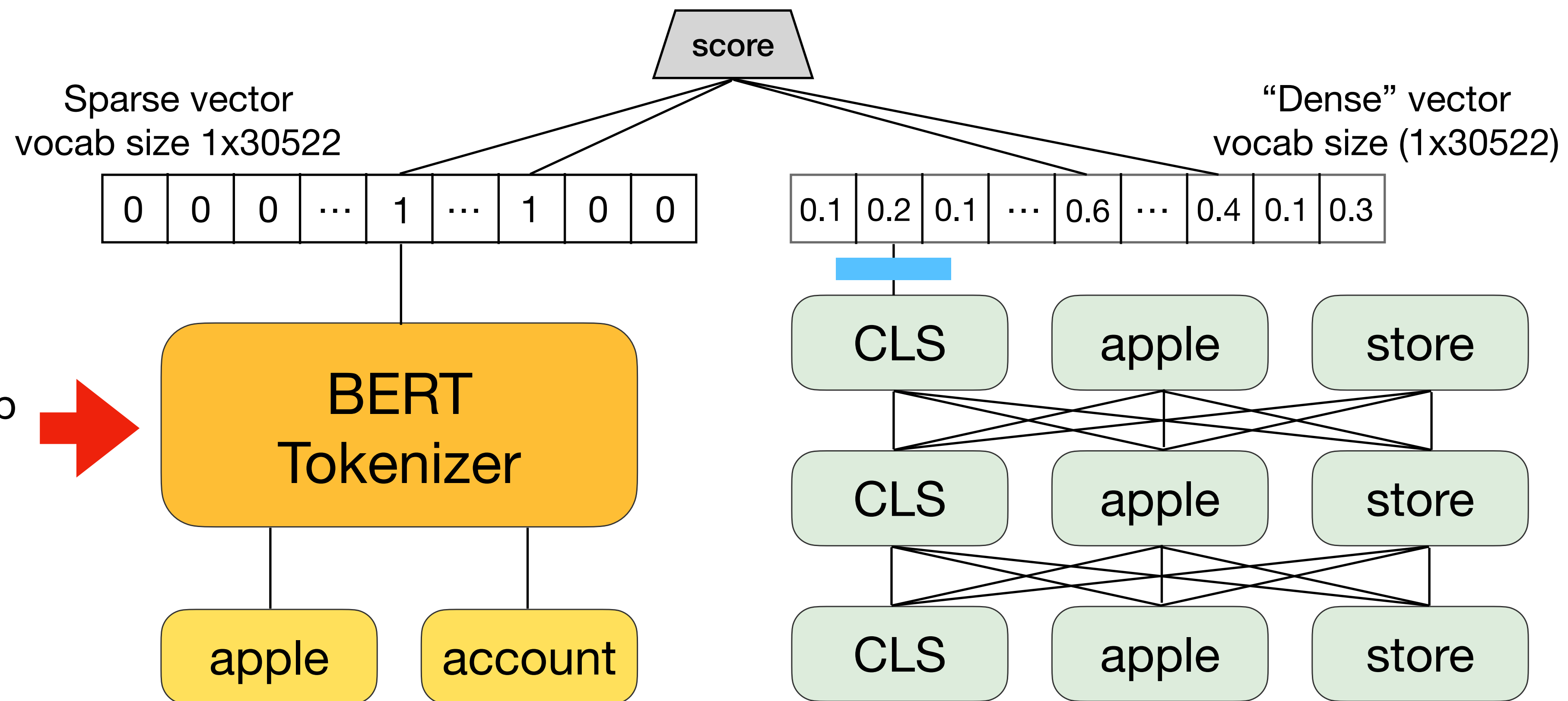
# TILDE



: query token



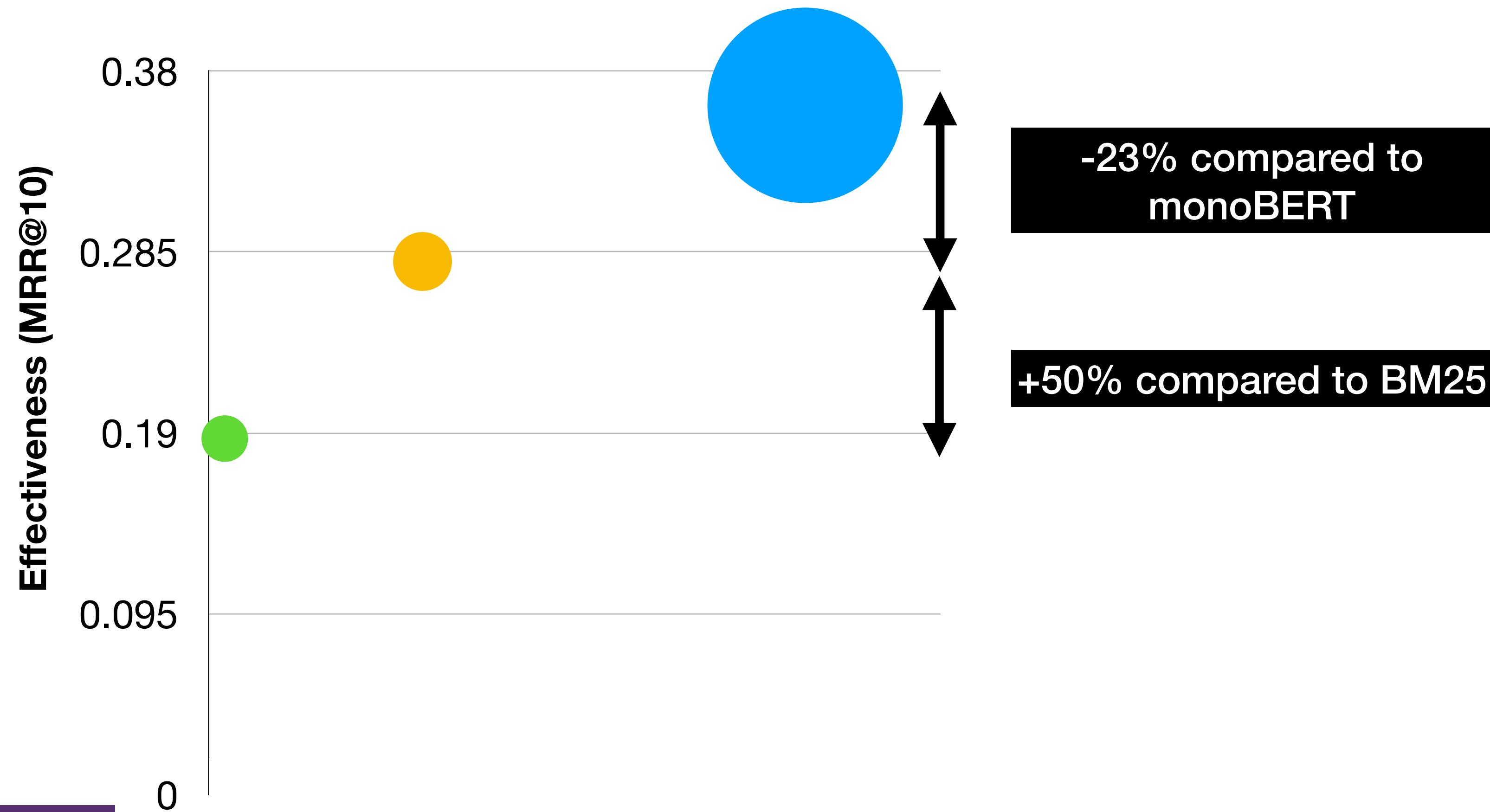
: passage token





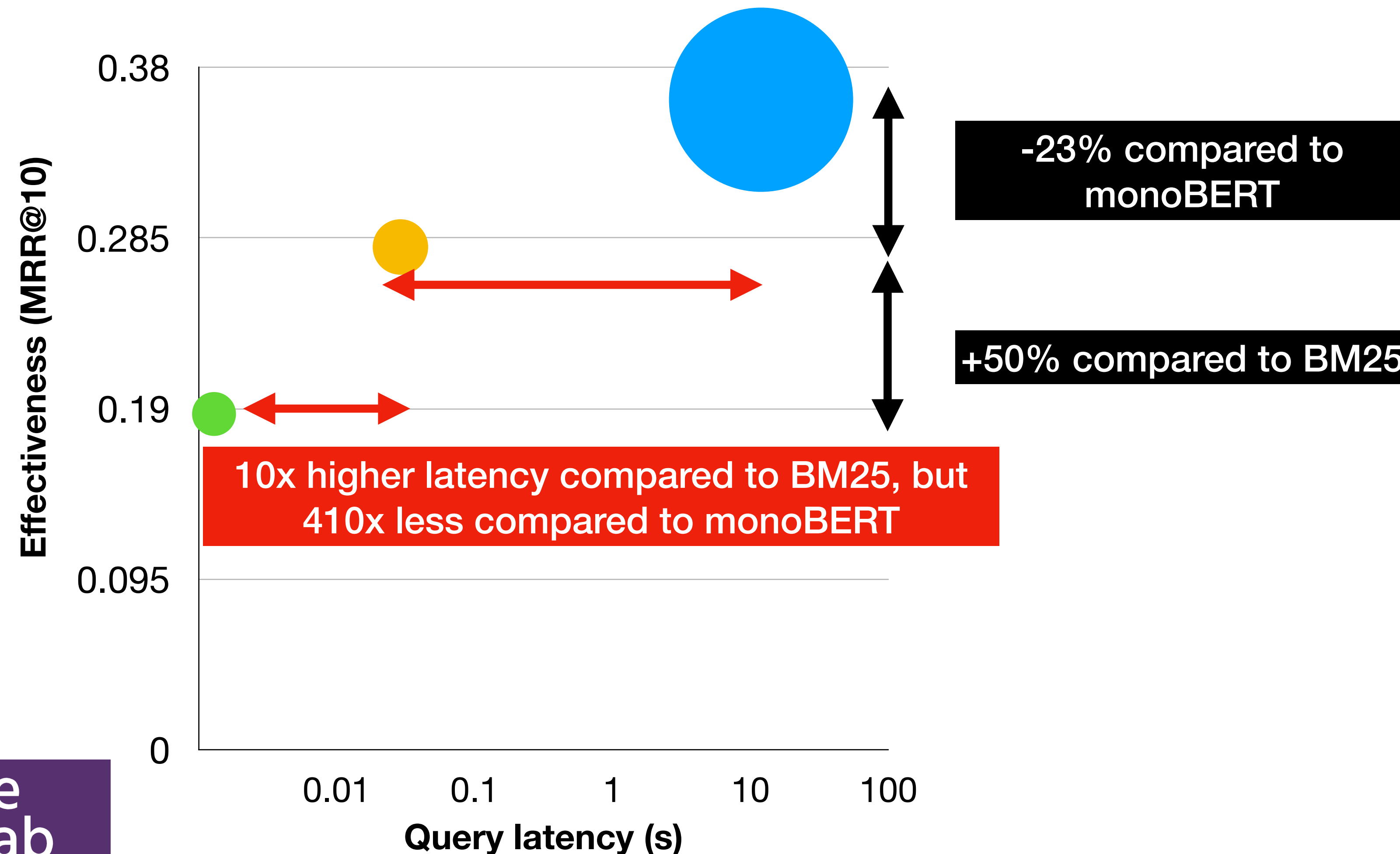
# Effectiveness/Efficiency/Hardware Trade-offs

● monoBERT    ● BM25  
● TILDE



# Effectiveness/Efficiency/Hardware Trade-offs

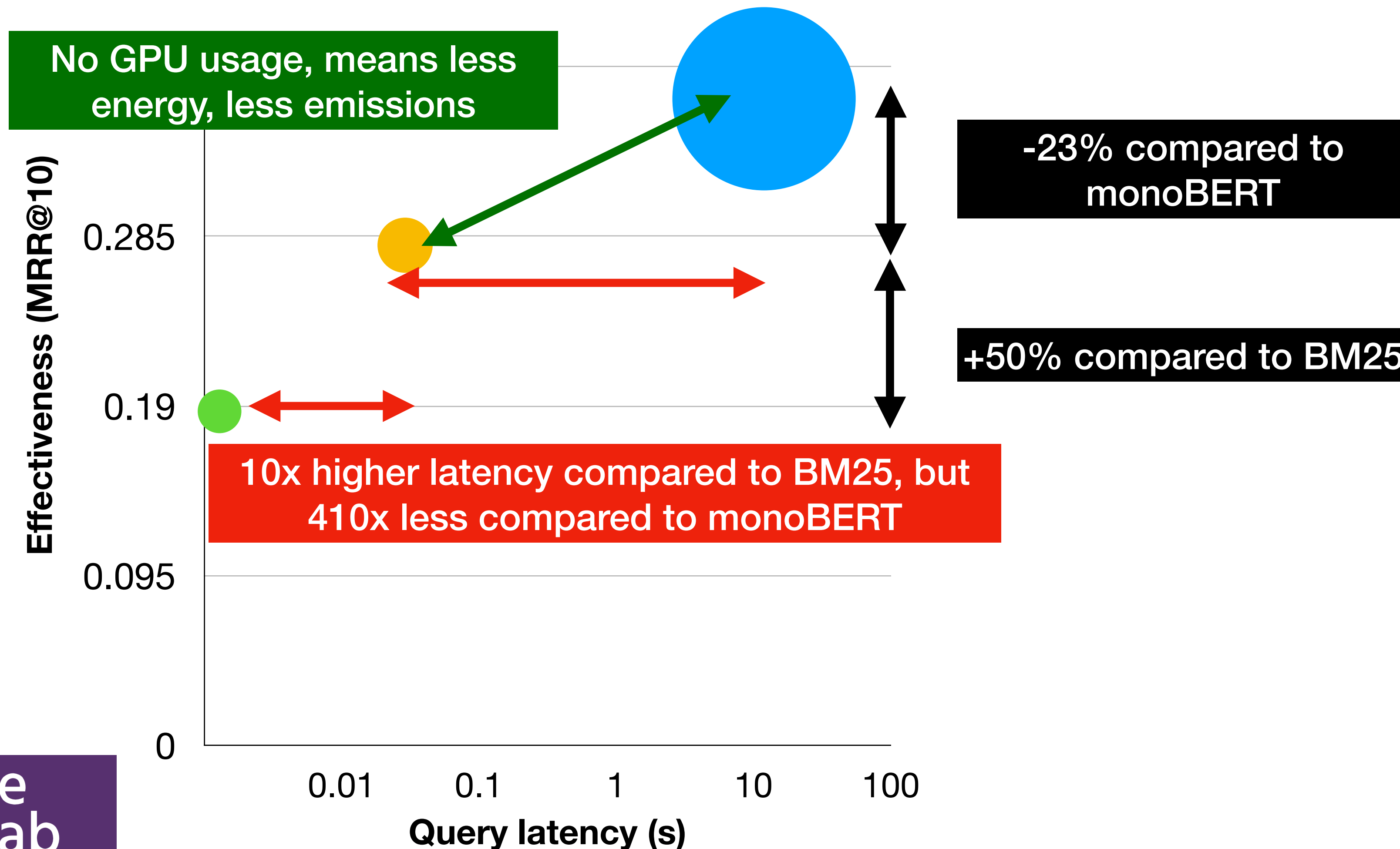
● monoBERT    ● BM25  
● TILDE





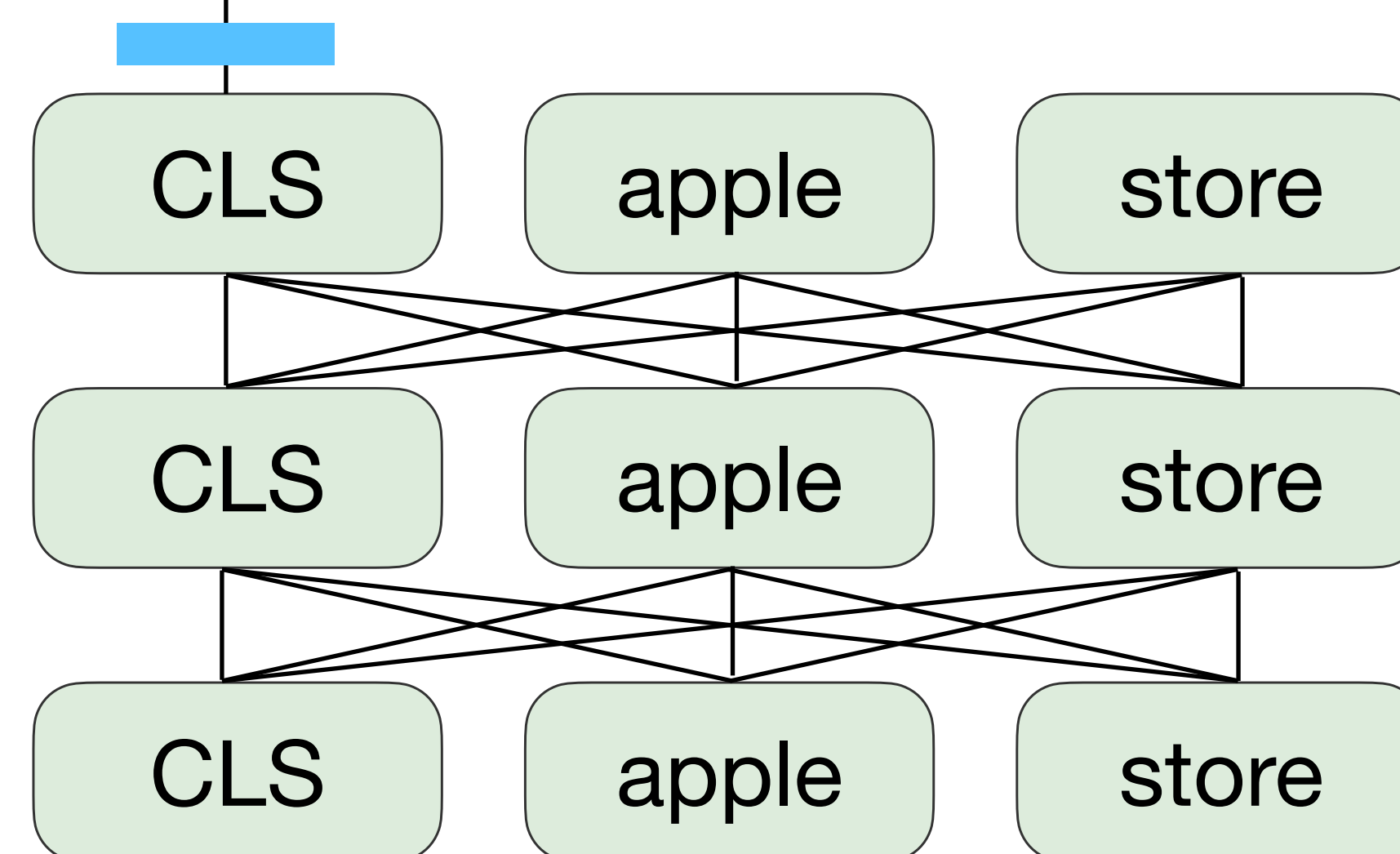
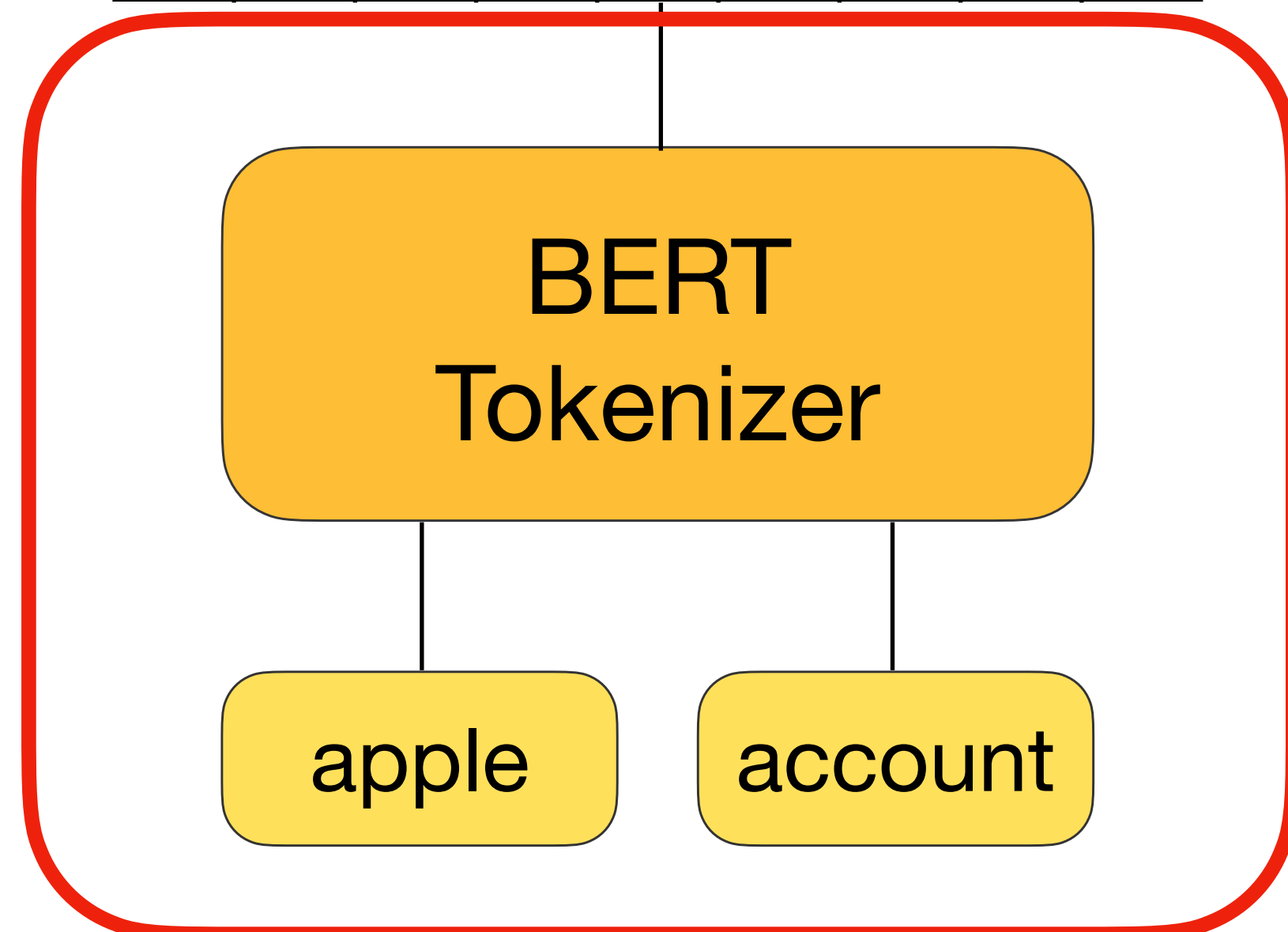
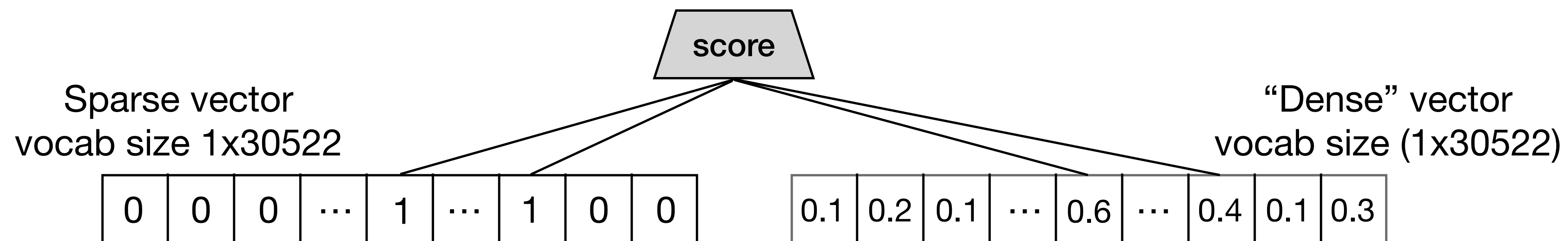
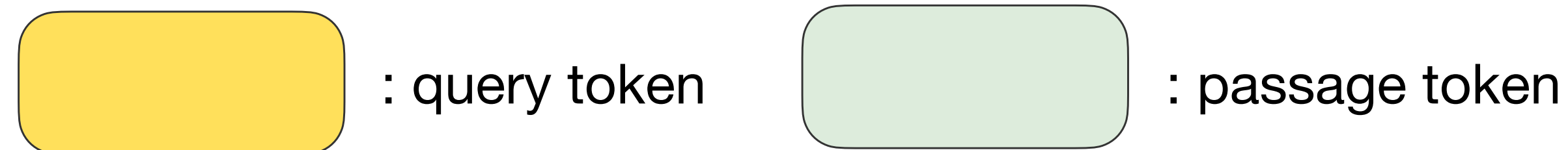
# Effectiveness/Efficiency/Hardware Trade-offs

● monoBERT    ● BM25  
● TILDE



- TILDE provides a trade-off between effectiveness and efficiency
- Does not require GPU at query time
- Yet, losses in effectiveness w.r.t. monoBERT are significant
- Less effective than DRs
- 10x less latency than DRs ran on GPU, 100x less latency than DRs ran on CPU

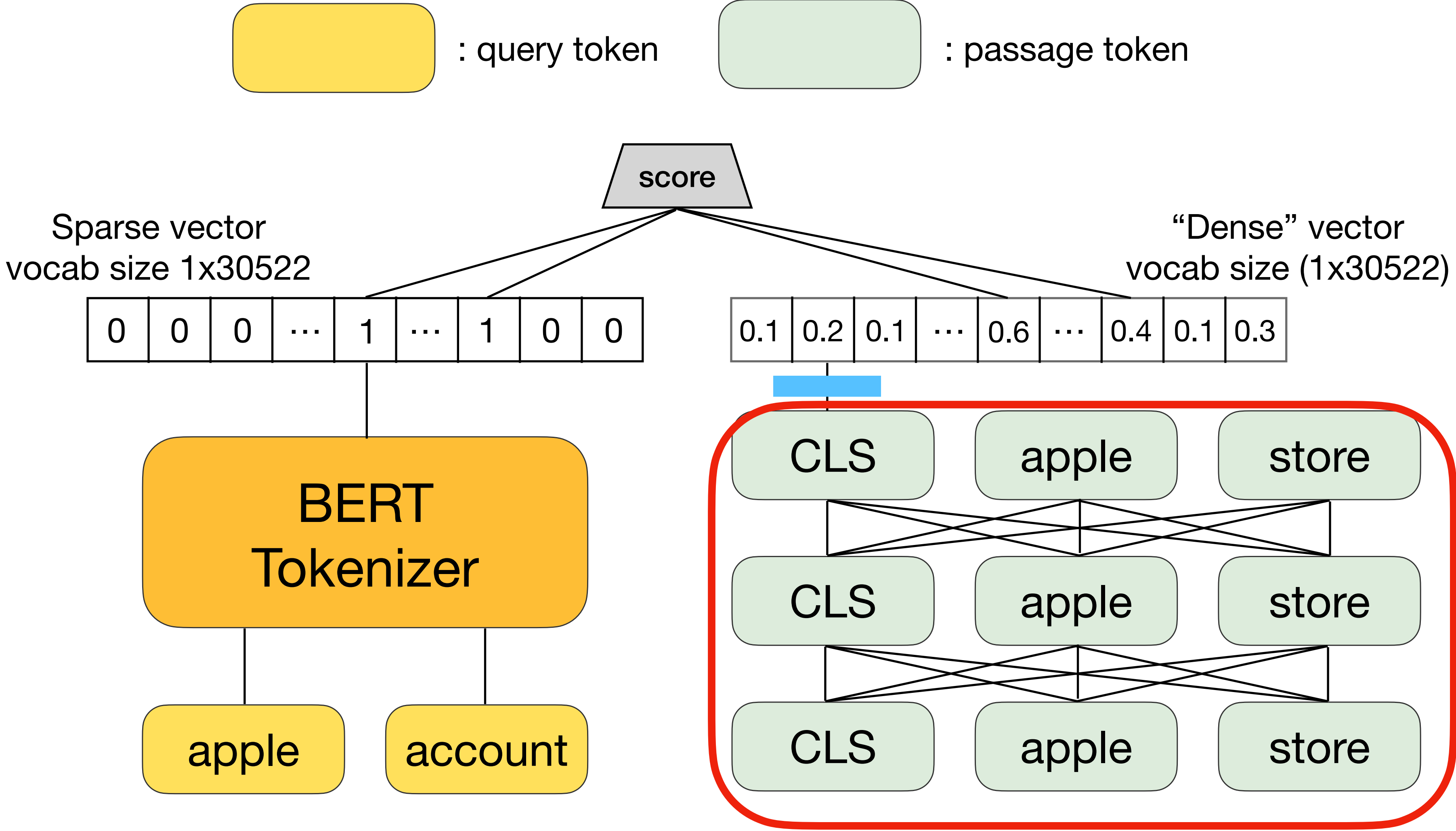
# Can we do any better w.r.t. efficiency & effectiveness?



**Already reduced to bare minimum; any increase of complexity means increased latency**

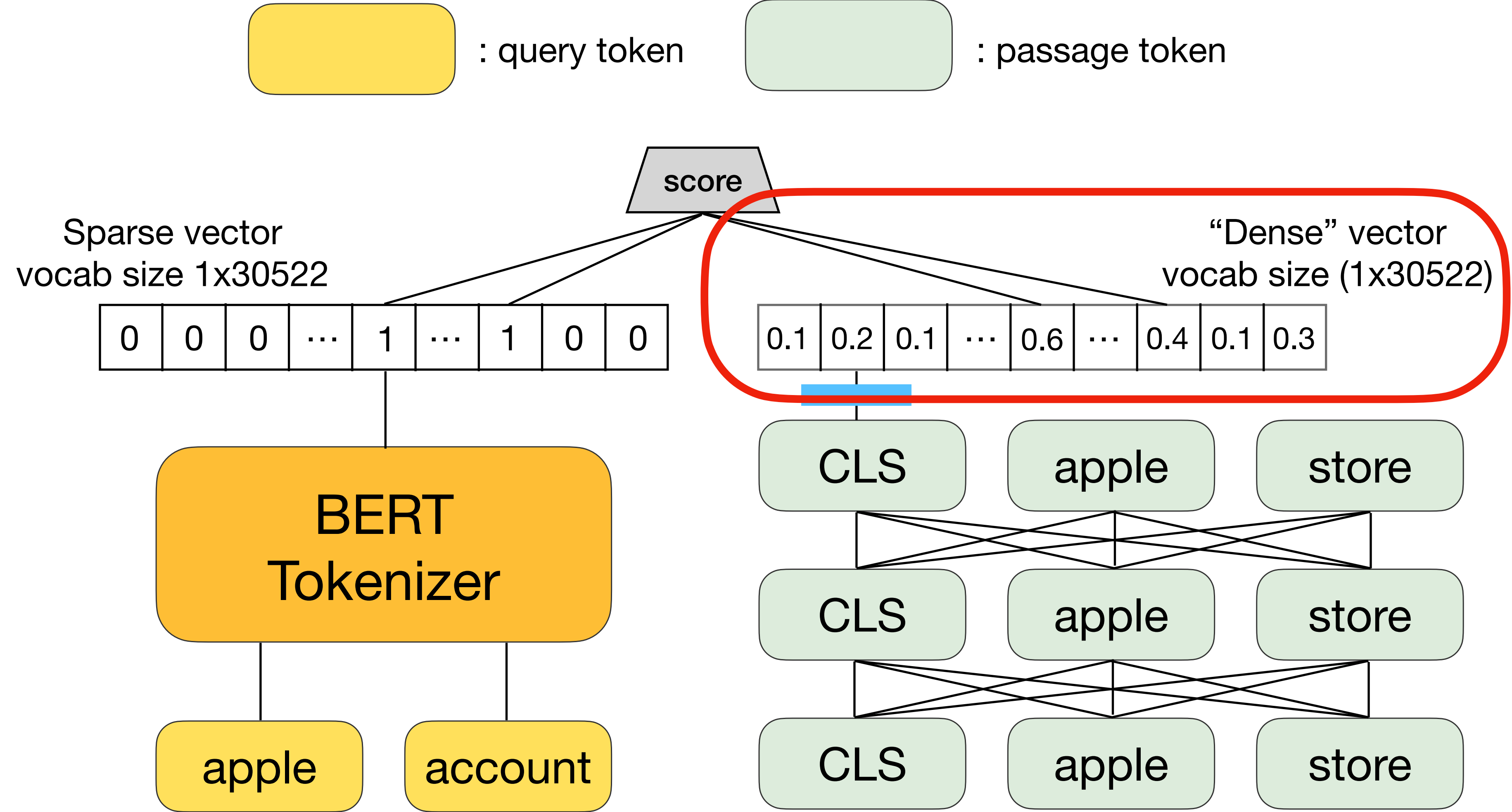


# Can we do any better w.r.t efficiency & effectiveness?



**Offline computation: increase of complexity does not affect latency. REFINE THIS!**

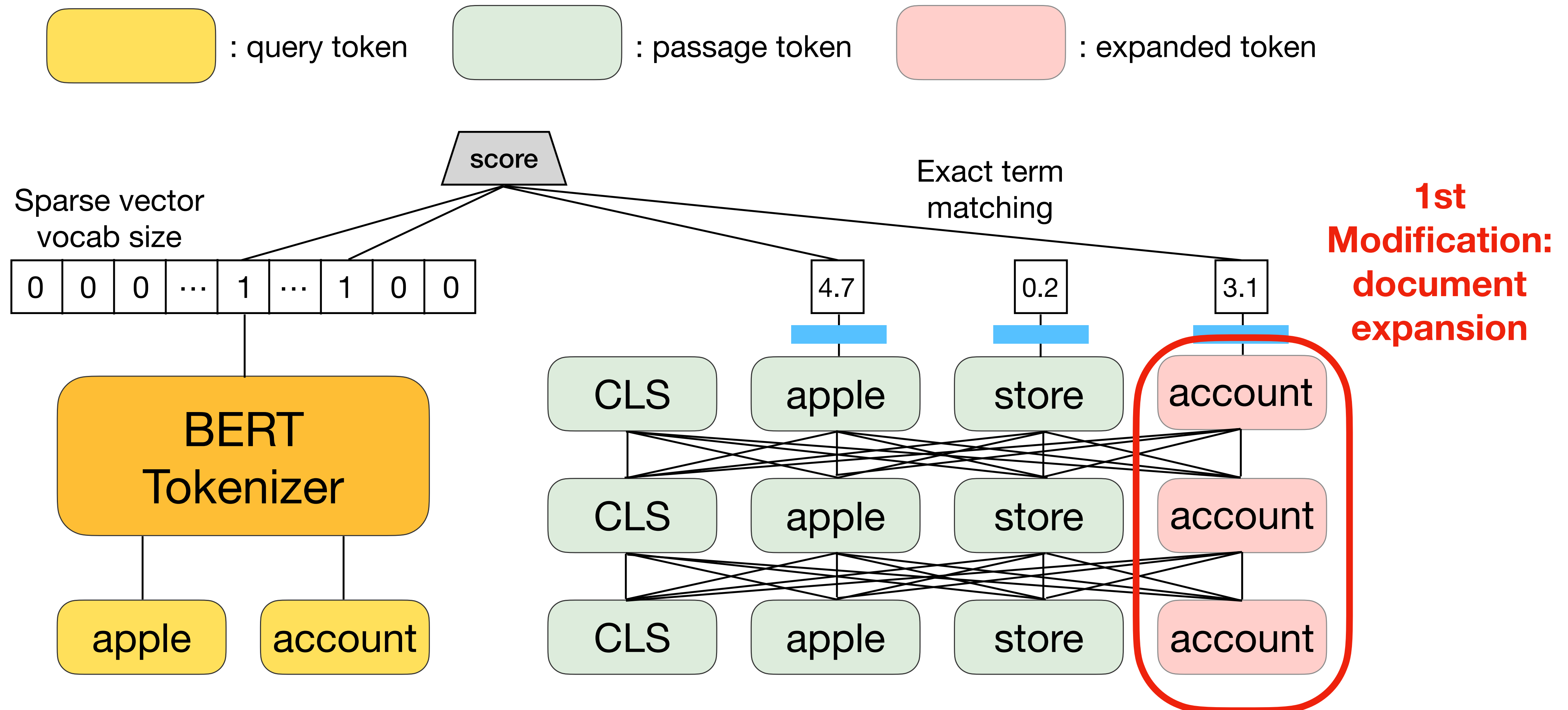
# Can we do any better w.r.t efficiency & effectiveness?



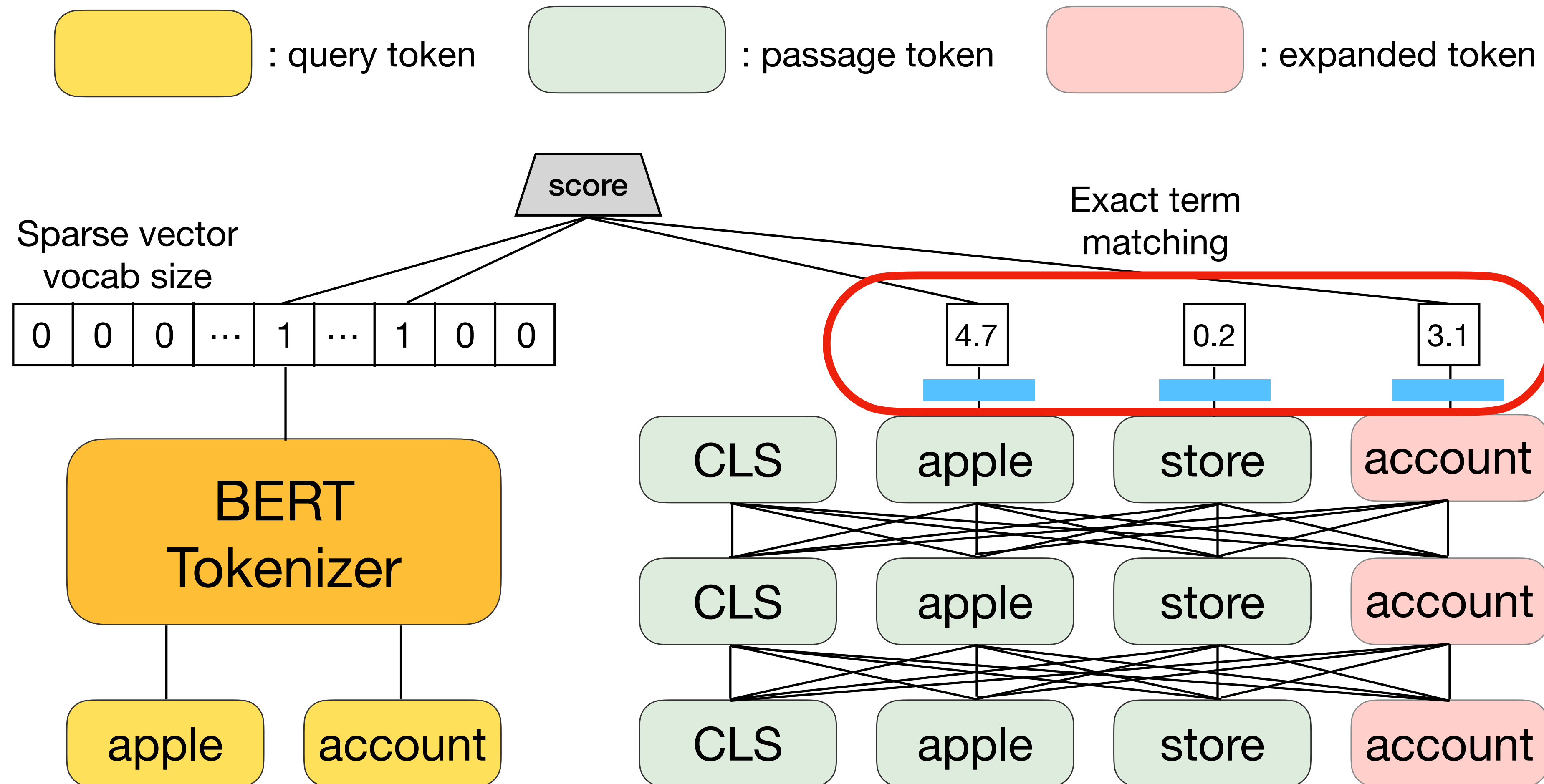
**Large vector  
to store.  
Additional  
projection, no  
direct relation  
to vocabulary.  
REFINE THIS!**



# TILDEv2



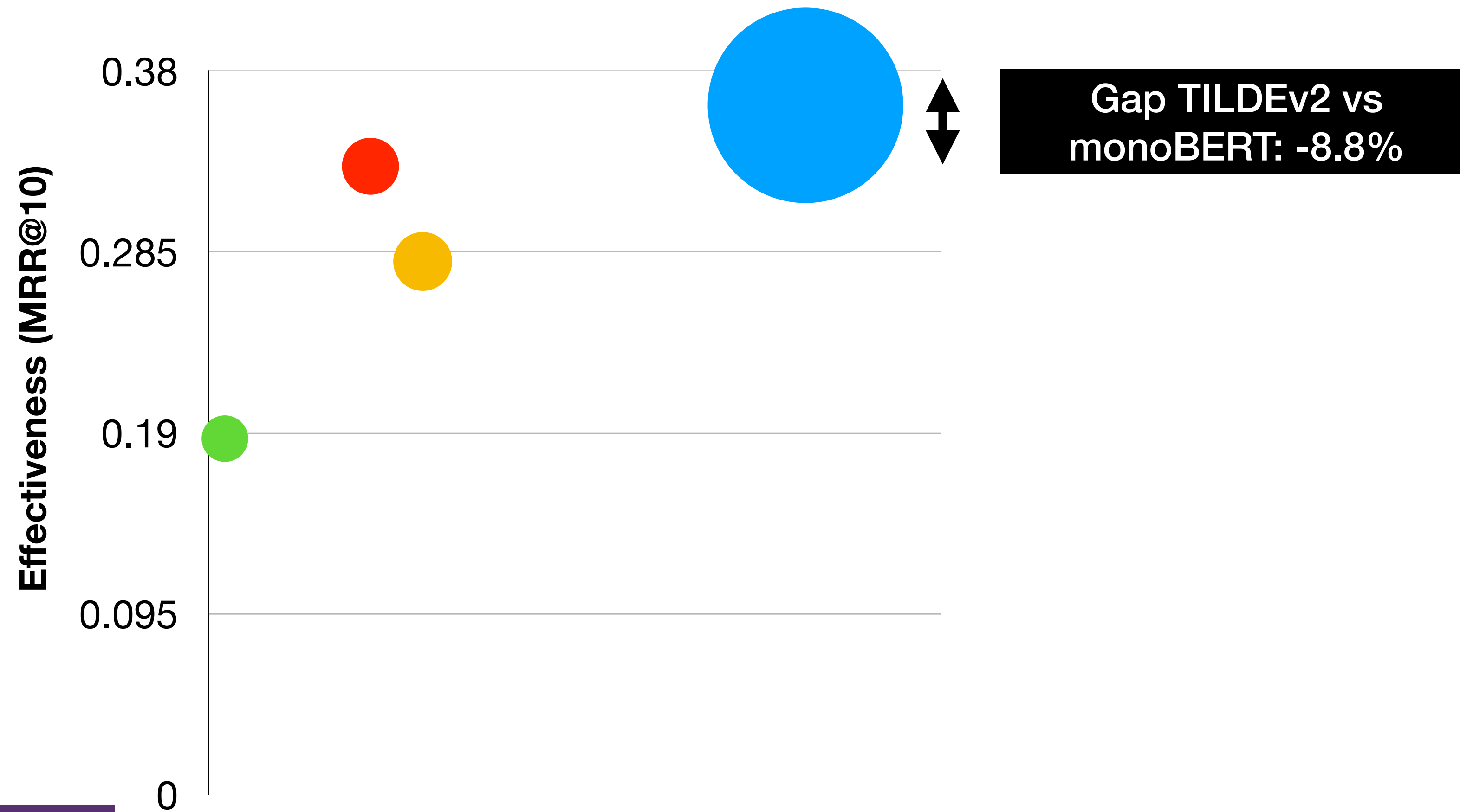
# TILDEv2



**2nd Modification:**  
projection token embedding into single score & exact matching with query

# Effectiveness/Efficiency/Hardware Trade-offs

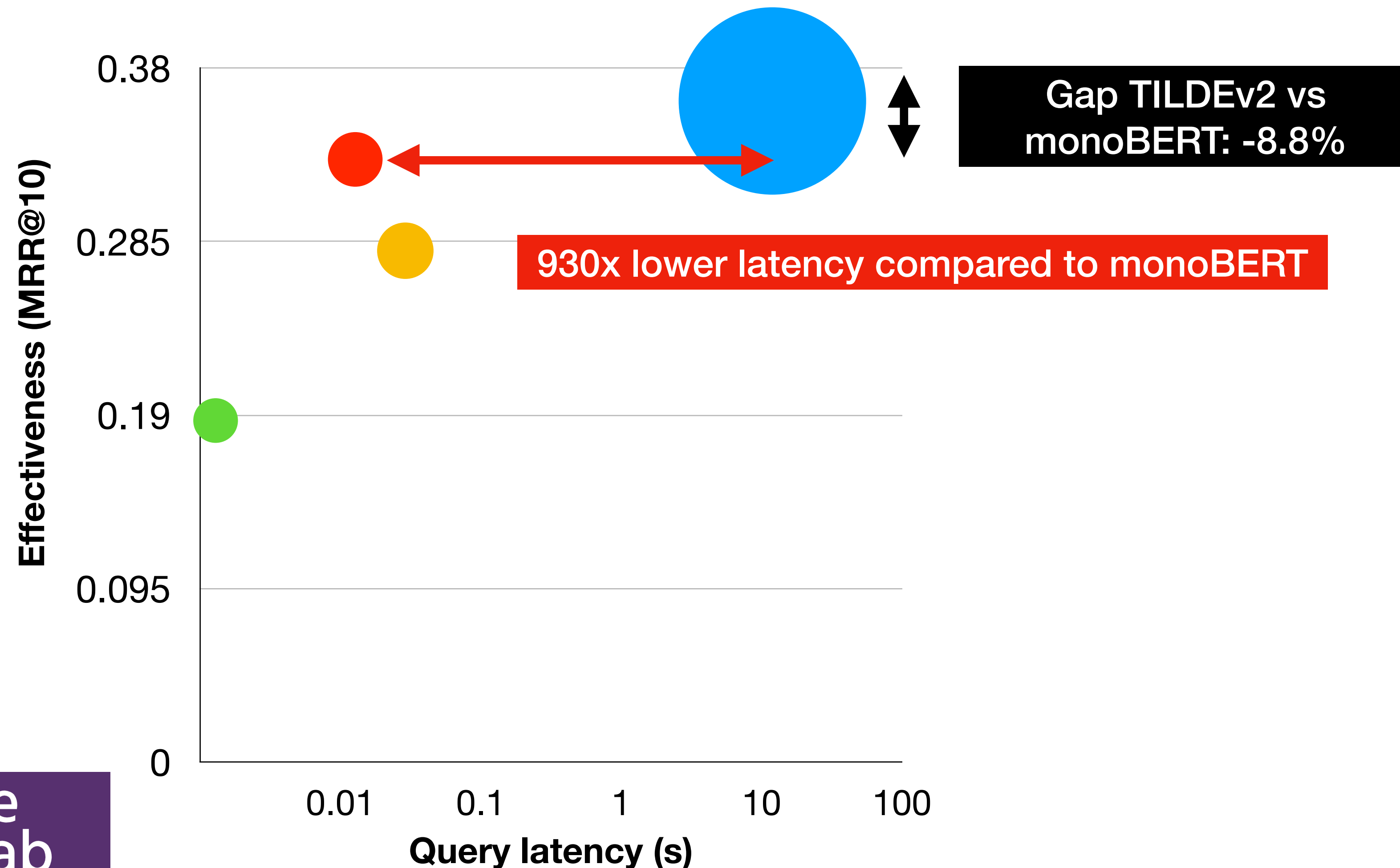
● monoBERT    ● BM25  
● TILDE    ● TILDEv2





# Effectiveness/Efficiency/Hardware Trade-offs

● monoBERT    ● BM25  
● TILDE       ● TILDEv2



- TILDEv2 achieved a great trade-off
- Can be run in production, on commodity hardware: it does not even require a GPU
- More effective & efficient than DRs
- Effectiveness at par to other sparse models (uniCOIL, SPLADE)

# Importance of Expansion in TILDEv2

	No Exp
MRR@10	0.299
Avg added tokens	-
Index size	4.3 GB

# Importance of Expansion in TILDEv2

	No Exp	Doc2query
MRR@10	0.299	0.333
Avg added tokens	-	19.0
Index size	4.3 GB	5.2 GB



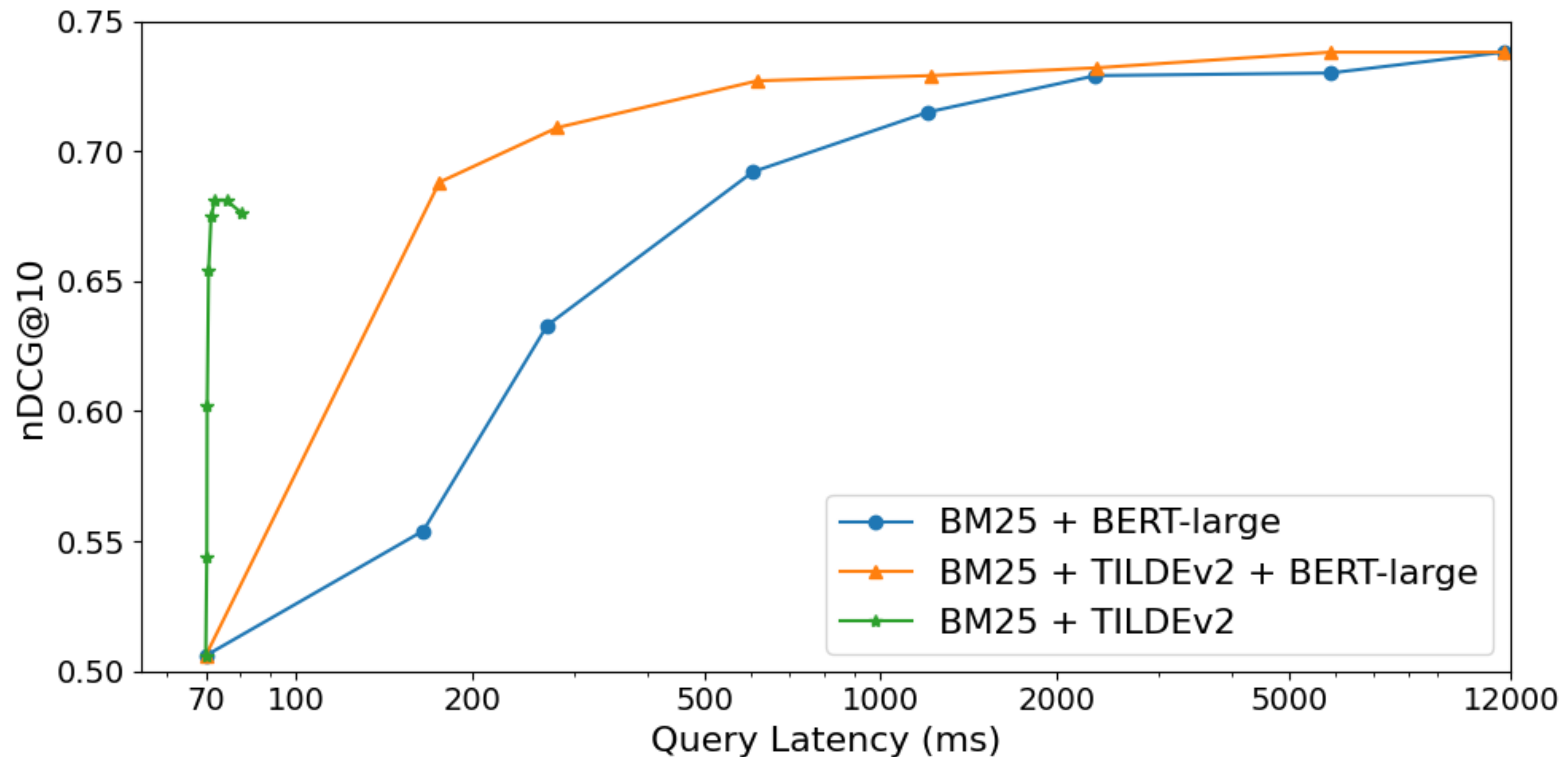
# Importance of Expansion in TILDEv2

	No Exp	Doc2query	TILDE m=128	TILDE m=150	TILDE m=200
MRR@10	0.299	0.333	0.326	0.327	0.330
Avg added tokens	-	19.0	13.0	25.2	61.6
Index size	4.3 GB	5.2 GB	5.2 GB	5.6 GB	6.9GB

# However, expansion comes at a cost

	No Exp	Doc2query	TILDE m=128	TILDE m=150	TILDE m=200
MRR@10	0.299	0.333	0.326	0.327	0.330
Avg added tokens	-	19.0	13.0	25.2	61.6
Index size	4.3 GB	5.2 GB	5.2 GB	5.6 GB	6.9GB
Expansion cost	-	320 hours, \$768	7.22 hours, \$5.34	7.25 hours, \$5.37	7.33 hours, \$5.42

# TILDEv2 Cost-Quality Trade-off

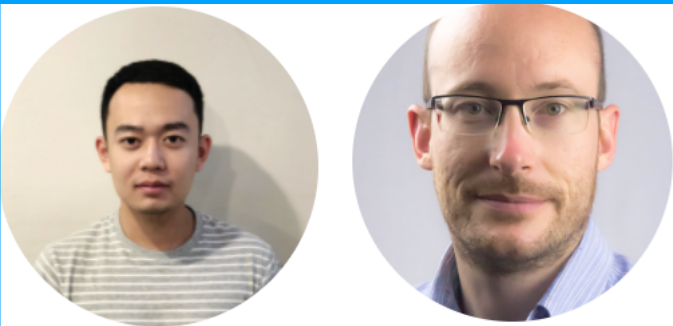




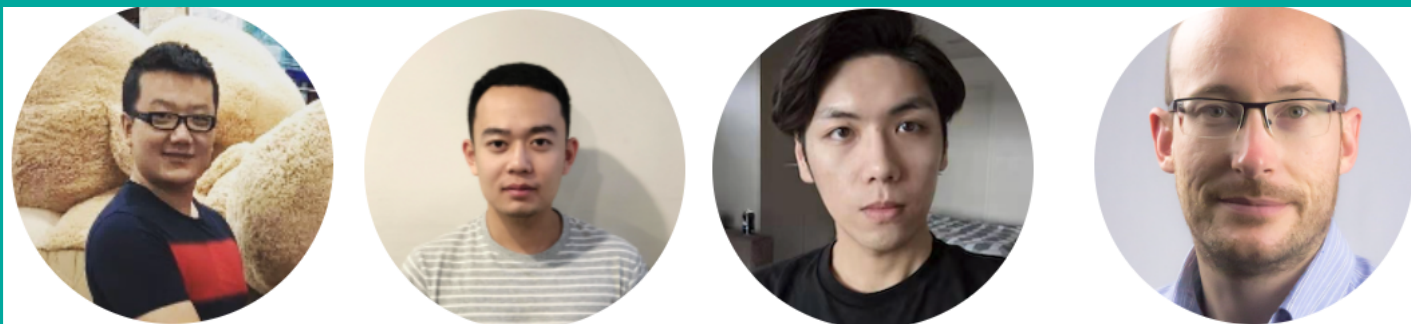
Trade-off between effectiveness,  
efficiency and hardware



Robustness to out-of-distribution  
data



PRF integration, efficient PRF



# Neural IR @ ielab

## Questions?

 @guidozuc

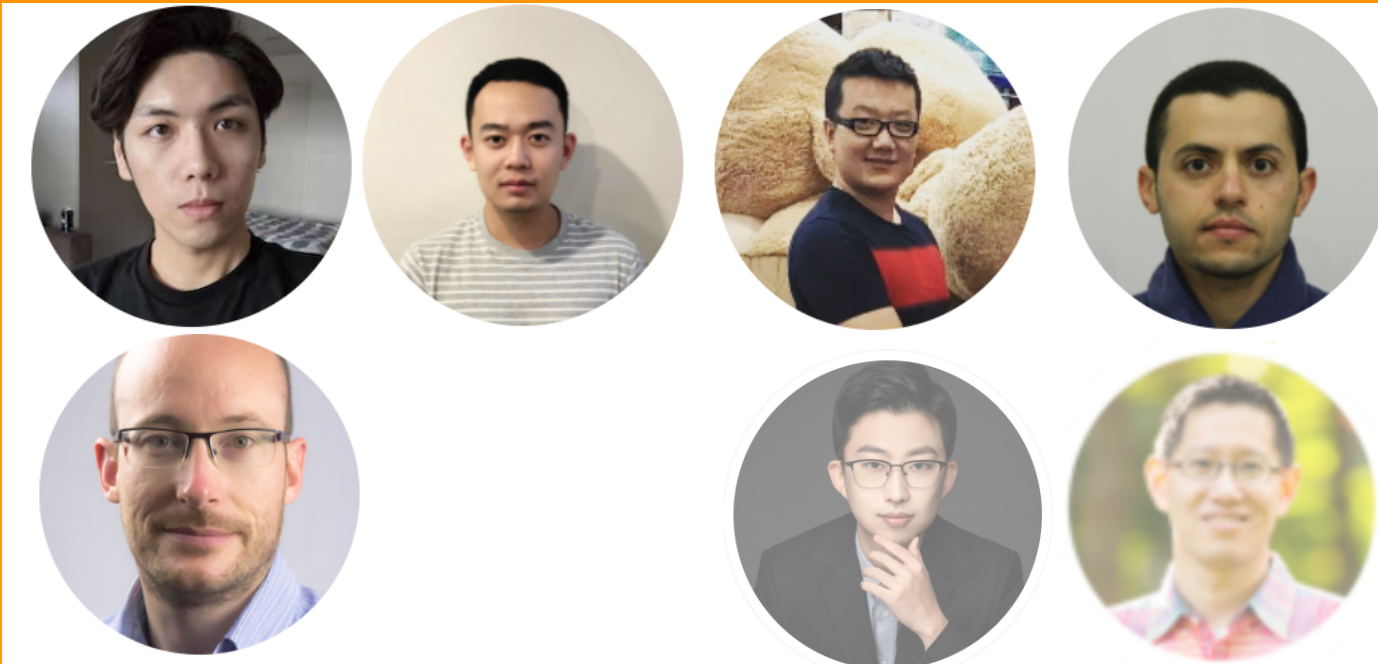
<https://ielab.io/guido>

[https://ielab.io/projects/  
transformers4ir.html](https://ielab.io/projects/transformers4ir.html)

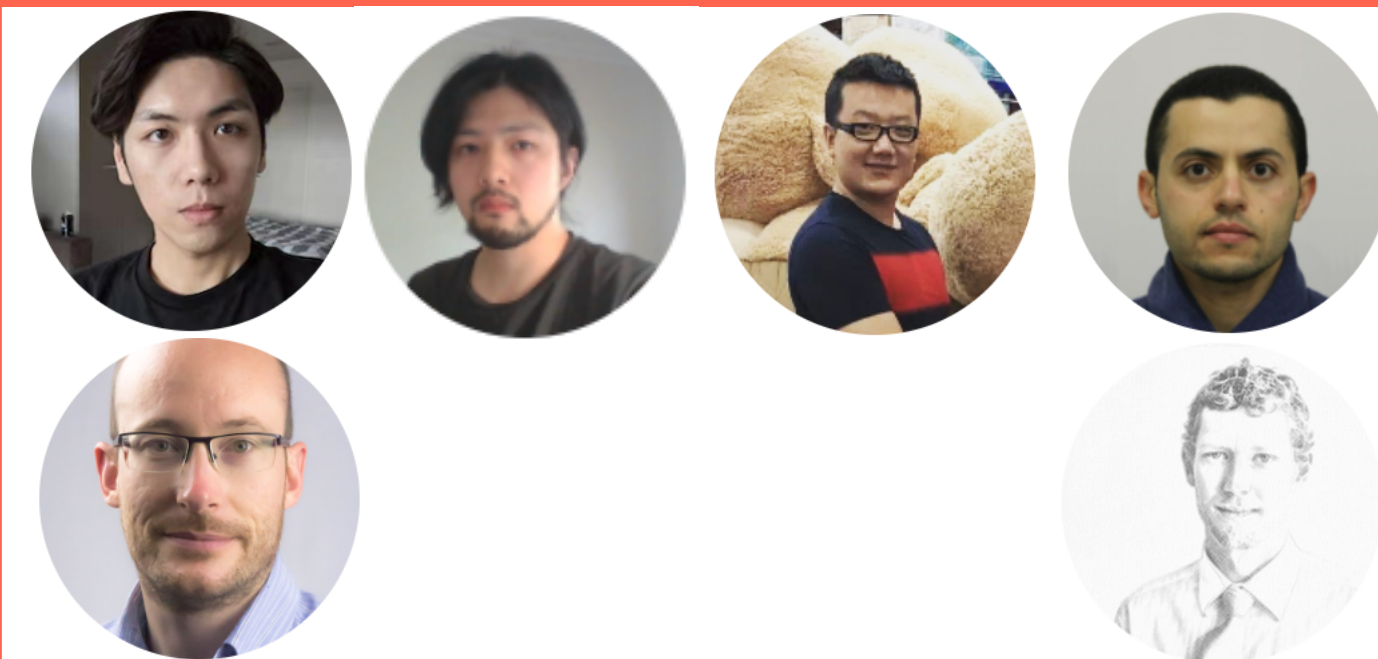
Data & Training efficiency



Hybrid sparse-dense methods



BERT models in domain-specific  
tasks





# Additional Material: Fast Passage Re-ranking with Contextualized Exact Term Matching and Efficient Passage Expansion

Shengyao Zhuang & Guido Zuccon


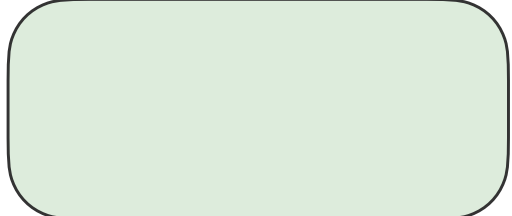
{s.zhuang,g.zuccon}@uq.edu.au

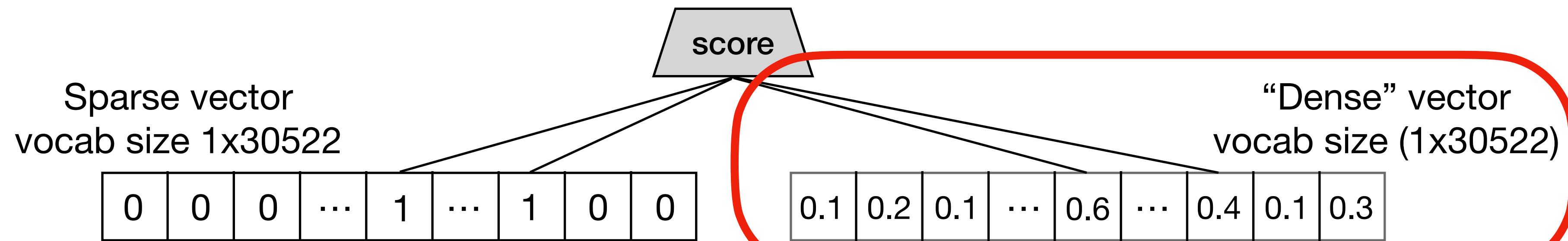
ielab, The University of Queensland, Australia

[www.ielab.io](http://www.ielab.io)



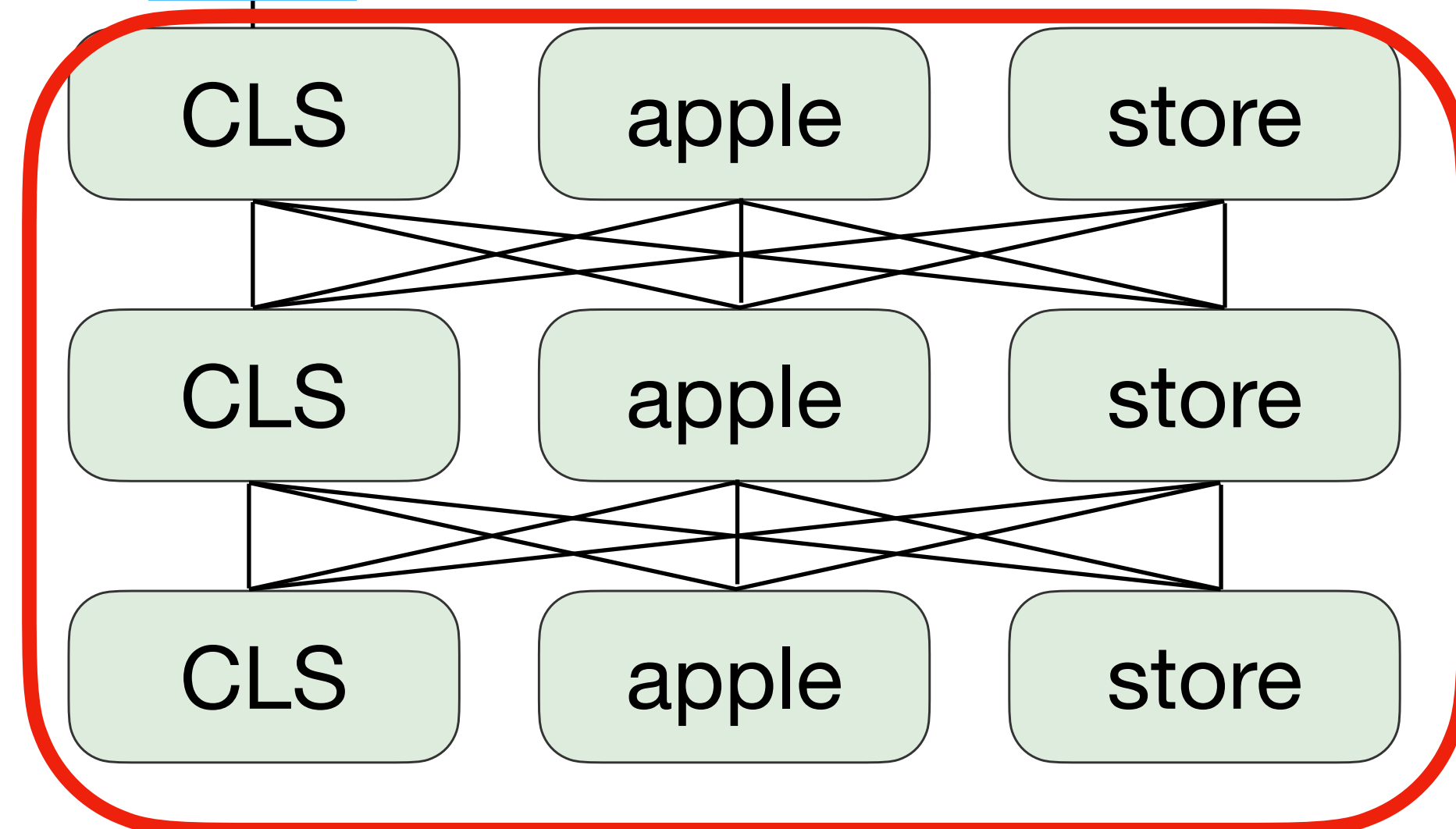
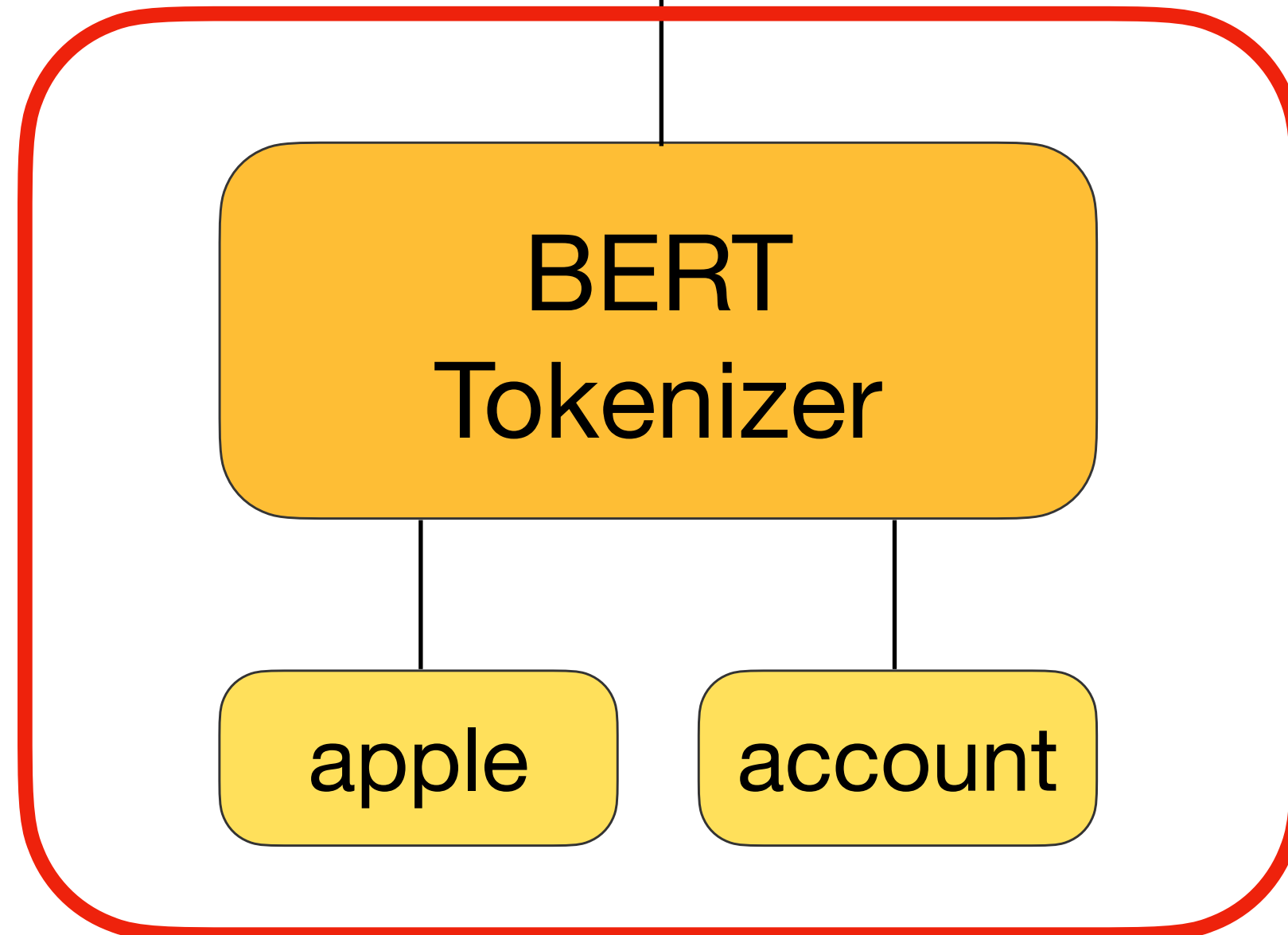
# Can we do any better w.r.t efficiency & effectiveness?

 : query token     : passage token



**Large vector to store.**  
**Additional projection, no direct relation to vocabulary.**  
**REFINE THIS!**

**Already reduced to bare minimum; any increase of complexity means increased latency**



**Offline computation: increase of complexity does not affect latency.**  
**REFINE THIS!**



# TILDEv2: extending TILDE with Contextualized Exact Term Matching and Passage Expansion

- Use BERT tokeniser to obtain sparse query encoding
- Use BERT token embeddings for exact term match
- Expand the document representation with doc2query or TILDE: improves the document representations, overcomes query-document vocabulary mismatch

# TILDE &TILDEv2 effectiveness

Method	MS MARCO Dev (MMR@10)	Latency GPU	Latency CPU
BM25 + monoBERT large	0.365	11,594	-
BM25 + monoBERT base	0.347	3,815	-
DPR	0.311		
RepBERT	0.304	152	1,633
ANCE	0.330	152	1,633
CLEAR	0.328		
EPIC	0.273	96	113
BM25+TILDE	0.269	-	76
doc2query+TILDE	0.285	-	75
BM25+TILDEv2	0.333	-	80
doc2query+TILDEv2	0.341	-	76

# Loss Function for TILDE

- Query tokens are assumed independent
- follow query/document likelihood ranking paradigm

$$\mathcal{L}_{QL}(D) = - \sum_{(q,d) \in D} \frac{1}{|V|} \sum_i^{|V|} y \log(P_{\theta}(t_i | d)) + (1 - y) \log(1 - P_{\theta}(t_i | d)),$$

Exploit the language model  
from observed documents

$$\mathcal{L}_{DL}(D) = - \sum_{(q,d) \in D} \frac{1}{|V|} \sum_i^{|V|} y \log(P_{\theta}(t_i | q)) + (1 - y) \log(1 - P_{\theta}(t_i | q)),$$

Exploit the language model  
from observed queries

$$\mathcal{L}_{BiQDL}(D) = \frac{\mathcal{L}_{QL}(D) + \mathcal{L}_{DL}(D)}{2}$$

Bi-directional query-  
document likelihood loss  
(BiQDL)

$$y = \begin{cases} 1, & \text{if } t_i \text{ in } q, \\ 0, & \text{otherwise.} \end{cases}$$

# 3 ways of ranking with TILDE

- TILDE can rank passages based on query likelihood only (TILDE-QL):
- TILDE can rank passages based on document likelihood only (TILDE-DL):
- TILDE can rank passages based on query and document likelihood (TILDE-QDL):

$$\text{TILDE-QL}(q|d^k) = \sum_i^{|q|} \log(P_\theta(q_i|d^k))$$

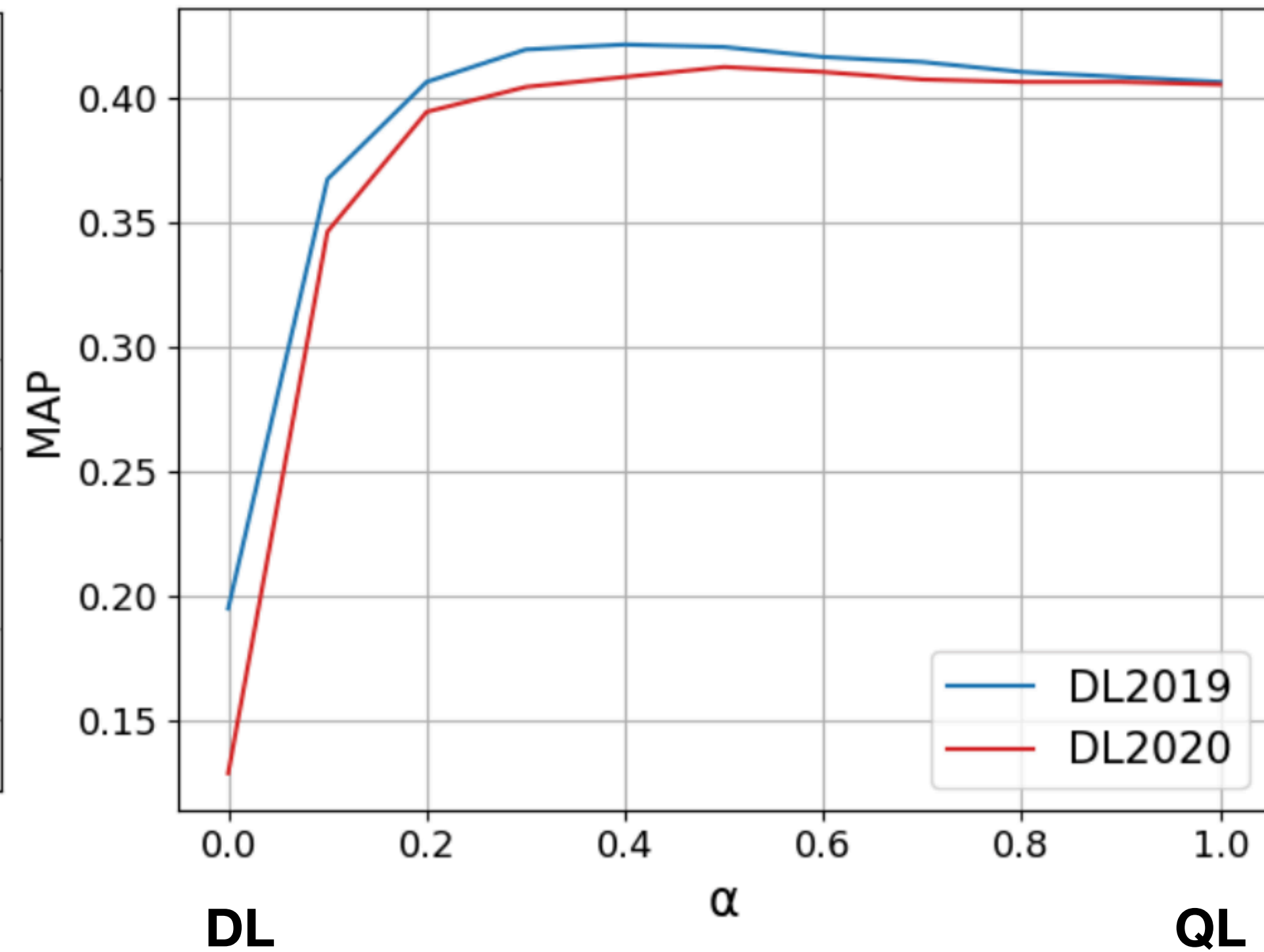
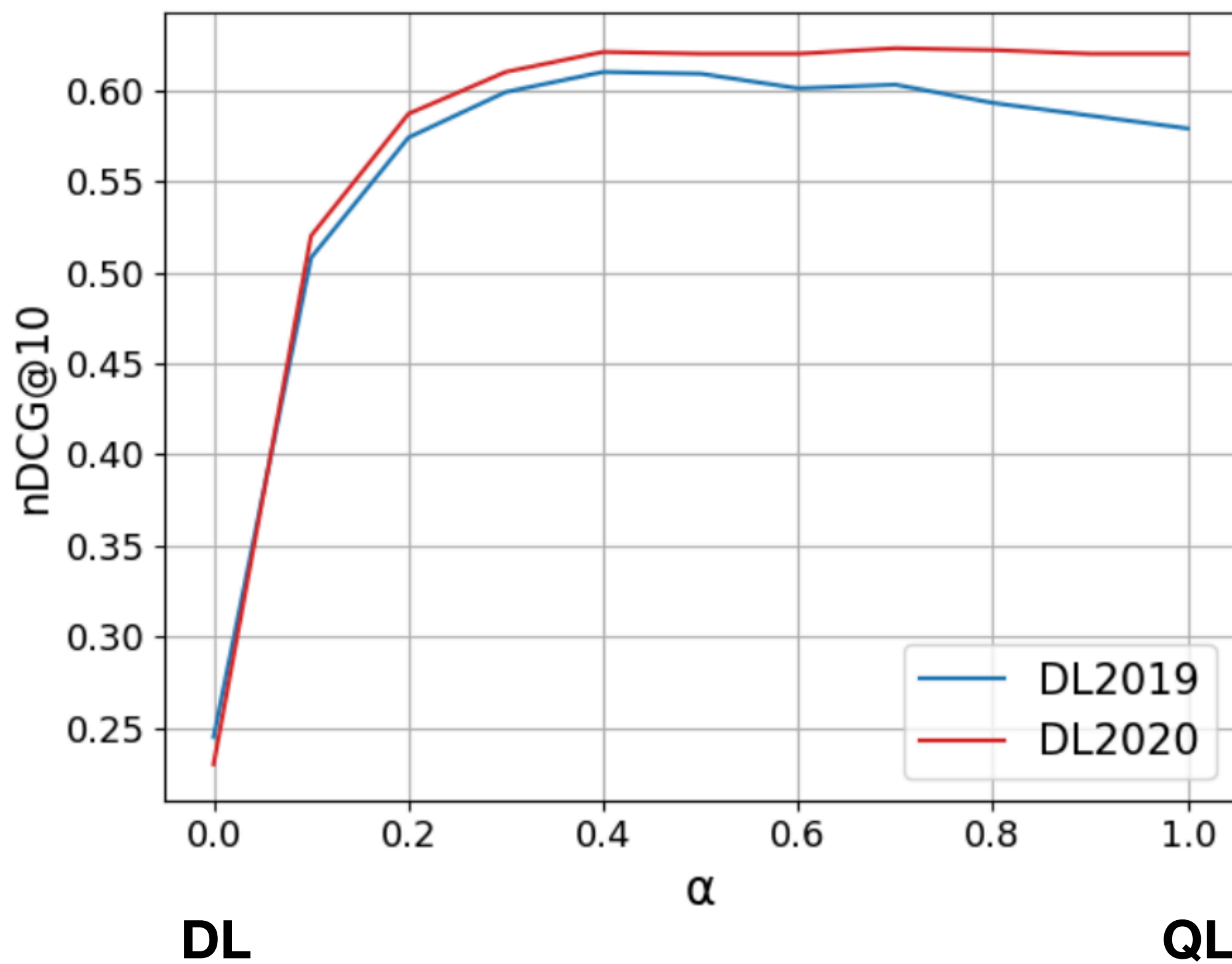
$$\text{TILDE-DL}(d^k|q) = \frac{1}{|d^k|} \sum_i^{|d^k|} \log(P_\theta(d_i^k|q))$$

$$\begin{aligned} \text{TILDE-QDL}(q, d^k) = \\ \alpha \cdot \text{TILDE-QL}(q|d^k) + (1 - \alpha) \cdot \text{TILDE-DL}(d^k|q) \end{aligned}$$

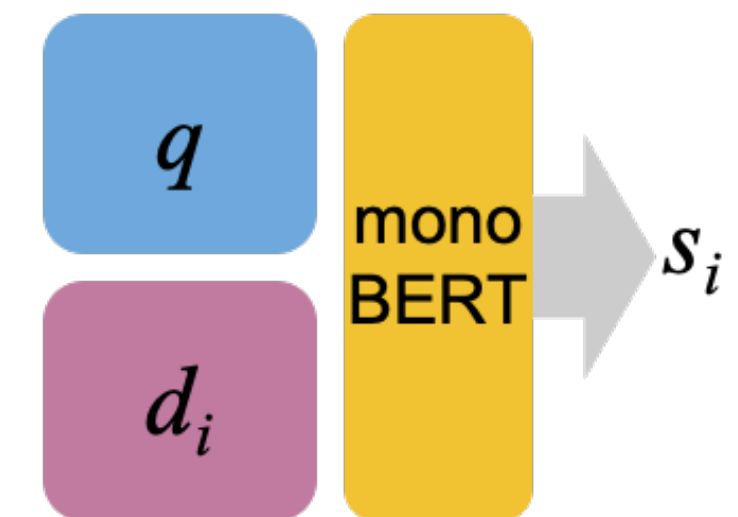


# Impact of document likelihood

$$\text{TILDE-QDL}(q, d^k) = \alpha \cdot \text{TILDE-QL}(q|d^k) + (1 - \alpha) \cdot \text{TILDE-DL}(d^k|q)$$



# Using monoBERT for ranking



# Using monoBERT for ranking

