To Case or Not to Case: An Empirical Study in Learned Sparse Retrieval

Emmanouil Georgios Lionis[1,2,3](✉), Jia-Huei Ju[2], Angelos Nalmpantis[3], Casper Thuis[3], Sean MacAvaney[1], and Andrew Yates[4]

[1] University of Glasgow, Glasgow, United Kingdom
e.lionis.1@research.gla.ac.uk
[2] University of Amsterdam, Amsterdam, The Netherlands
[3] TKH AI, Amsterdam, The Netherlands
[4] Johns Hopkins University, Baltimore, USA

Abstract. Learned Sparse Retrieval (LSR) methods construct sparse lexical representations of queries and documents that can be efficiently searched using inverted indexes. Existing LSR approaches have relied almost exclusively on uncased backbone models, whose vocabularies exclude case-sensitive distinctions, thereby reducing vocabulary mismatch. However, the most recent state-of-the-art language models are only available in cased versions. Despite this shift, the impact of backbone model casing on LSR has not been studied, potentially posing a risk to the viability of the method going forward. To fill this gap, we systematically evaluate paired cased and uncased versions of the same backbone models across multiple datasets to assess their suitability for LSR. Our findings show that LSR models with cased backbone models by default perform substantially worse than their uncased counterparts; however, this gap can be eliminated by pre-processing the text to lowercase. Moreover, our token-level analysis reveals that, under lowercasing, cased models almost entirely suppress cased vocabulary items and behave effectively as uncased models, explaining their restored performance. This result broadens the applicability of recent cased models to the LSR setting and facilitates the integration of stronger backbone architectures into sparse retrieval. The complete code and implementation for this project are available at: https://github.com/lionisakis/Uncased-vs-cased-models-in-LSR.

Keywords: Document retrieval · Learned Sparse Retrieval · Cased and Uncased models

1 Introduction

Ad hoc retrieval, a central task in Information Retrieval (IR) [4], aims to return the most relevant documents in response to user queries. Traditional IR methods, such as TF-IDF [13] and BM25 [33], rely on statistical feature extraction and exact lexical matching between query terms and document content. These

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2026
R. Campos et al. (Eds.): ECIR 2026, LNCS 16483, pp. 512–528, 2026.
https://doi.org/10.1007/978-3-032-21289-4_33

approaches form the backbone of efficient retrieval systems, like PISA [23], offering both scalability and strong performance. However, because they depend on exact term matching, inconsistencies can arise when terms differ due to surface-level characteristics, such as inflection and capitalization. Most IR pipelines apply pre-processing steps such as stemming and lowercasing to mismatches between query and document terms [1,16].

Recent advances in transformer-based pre-trained language models have transformed natural language understanding [7], enabling retrieval systems to move beyond lexical overlap toward richer semantic matching. This shift has spurred the development of Neural Information Retrieval (Neural IR) [11,12]. Within this paradigm, **Learned Sparse Retrieval** (LSR) [10,26,39,43] has emerged as a particularly effective approach. LSR leverages the semantic knowledge of pre-trained models to assign learned weights to lexical terms, encoding queries and documents as sparse vectors over a fixed vocabulary. While most vector dimensions have a weight of zero, the non-zero values capture term importance, analogous to concepts such as term frequency or TF-IDF. This representation preserves compatibility with inverted index infrastructures [19,22,24] while offering greater transparency by explicitly highlighting the terms that contribute to relevance scoring.

Built on top of pre-trained language models, the effectiveness of LSR methods is closely tied to the specific model variant employed. Most early pre-trained models were released in both *cased* and *uncased* forms [7,34]. Cased models preserve capitalization, distinguishing tokens such as *Apple* and *apple*. This is advantageous in tasks, such as Named Entity Recognition (NER) [18], where case is an important signal, often influencing semantic meaning. In contrast, uncased models convert all text to lowercase, merging both forms into a single token. This approach reduces data sparsity and aligns with the lowercasing practices common in traditional IR pipelines [1,16], making it effective for contexts where case distinctions are less critical (e.g., open-domain question answering [14]). However, removing casing also discards potentially meaningful differences, such as those between the company *Apple* and the fruit *apple*.

Despite these trade-offs, prior work in LSR [10,26] has predominantly relied on uncased models, with limited systematic investigation into how casing choices affect retrieval effectiveness. In contrast, several recent studies [8,30,38,40] have explored cased models, specifically decoders such as T5 [31] and OPT [41], without considering their uncased counterparts. This leaves an important gap in understanding the behavior of cased versus uncased models and complicates the use of state-of-the-art architectures, such as ModernBERT [37], which are released only in cased forms. These practical challenges underscore the need for a careful examination of casing in LSR. Given the strong influence of model vocabulary on the output representations of LSR models, coupled with the language modeling community's increasing shift toward cased-only models, it is important to systematically investigate the role of casing to ensure the long-term viability of LSR methods.

In this work, we aim to address this gap by investigating the following research questions (RQs):

RQ1: How does the casing of backbone models impact the performance of LSR?
RQ2: Can lowercasing pre-processing improve the performance of cased models?
RQ3: Can post-processing strategies, such as constraining logits or modifying training to reduce capitalization reliance, improve cased models?
RQ4: What is the effect of backbone model casing on zero-shot task transfer?

Our experiments demonstrate that the choice of cased versus uncased models has a substantial impact on LSR performance. Specifically, uncased models generally outperform cased models when no pre-processing is applied. However, applying lowercasing pre-processing to cased models effectively mitigates this gap, enabling them to achieve performance comparable to uncased models. Post-processing techniques, such as using only the uncased tokens or regularizing cased tokens, provide modest efficiency gains while causing only a slight decrease in accuracy. Additionally, we analyze the distribution of token types (cased vs. uncased) in model outputs based on the input token type. The results show that, without pre-processing, cased models frequently map cased inputs to uncased tokens, indicating a systematic preference for the uncased vocabulary. With lowercasing, cased models rely almost entirely on uncased tokens, effectively mirroring the behavior of uncased models. These findings demonstrate that while casing introduces challenges for LSR, appropriate pre-processing enables cased models to remain competitive and effective for both in-domain and out-of-domain retrieval tasks.

2 Related Work

2.1 Learned Sparse Retrieval

LSR methods encode queries and documents into sparse representations with two encoders, a query encoder (f_Q) and a document encoder (f_D). These two encoders, which can be the same or distinct, encode text as a high-dimensional sparse vector with a dimension $|V|$ (matching the size of the model's vocabulary, typically around 30k) where the majority of the elements are zero.[1] Document relevance is computed using the dot product between the query and document vectors:

$$score(q, d) = f_Q(q) \cdot f_D(d) = w_q \cdot w_d = \sum_{i=1}^{|V|} w_q^i w_d^i, \tag{1}$$

where w^i denotes the predicted weight for the ith term in the vocabulary.

[1] To induce this sparsity in the output representations, there have been numerous pruning and regularization methods, like top-k pruning [20], FLOPS regularization [28], and DF-FLOPS regularization [29].

The primary distinction between LSR and traditional methods like BM25 lies in their approach to tokenization and term weighting. In terms of tokenization, traditional lexical models use a rule-based pre-processing pipeline (usually including steps like stopword removal, stemming, and lowercasing) while LSR usually uses the backbone language model's tokenizer (usually a WordPiece [35] or BPE [36] model). For term weighting, BM25 relies on corpus-level statistics (e.g., TF-IDF) and does not require training, while LSR learns term weights through semantic processing and supervised training. Nonetheless, both approaches produce term-based representations, making LSR compatible with existing infrastructure, such as the inverted indexes originally developed for BM25 [5].

SPLADE [10], a state-of-the-art method, leverages a single pretrained model as an encoder for both query and document by using the masked language modeling (MLM) head of the backbone model. This approach performs term expansion and term weighting in an end-to-end manner. The encoder transforms an input sequence t (either query or document) into a logit matrix $\mathbf{W}$, where $W_{i,j}$ represents the weight of the ith term in the input sequence for the jth term in the vocabulary. The output provides a sparse representation for each term in the sequence, assigning a non-negative log-scaled weight to each vocabulary term. This representation is then reduced through column-wise max-pooling, selecting the maximum logit value for each term across the sequence. The term weights are formulated as:

$$w_i(t) = \log\left(1 + \max_{j=1}^{L} \text{ReLU}\left(h_j^\top e_i + b_i\right)\right) \tag{2}$$

where ReLU ensures non-negativity and the log normalization prevents excessive weight values.

2.2 Cased vs. Uncased Models

The choice between cased and uncased pre-trained models has been widely studied in NLP. BERT [7] was originally released in both cased and uncased versions, with the recommendation to use the cased variant for tasks where capitalization carries semantic information (e.g., NER), and the uncased variant for more general applications such as open-domain question answering.

In the early stages of Neural IR, both dense methods (e.g., DPR [6], ColBERT [15]) and sparse methods (e.g., SPARTA [42], SPLADE [10]) predominantly employed uncased versions of models such as BERT and DistilBERT, largely due to the focus on open-domain document retrieval where benchmarks like MS MARCO [27] were widely evaluated. Despite their prevalence, no explicit comparisons between cased and uncased models were conducted in IR, and to the best of our knowledge, no systematic evaluation has been reported, leaving uncased models as the dominant choice. By contrast, NLP research has investigated this issue more directly; for example, [17] showed that BERT-cased can achieve sufficient performance in open-ended question answering when preprocessing is applied.

Cased models have also been shown to be beneficial in domain-specific text mining tasks where capitalization encodes critical semantics. This is observed by BioBERT [18], which leverages cased inputs to achieve superior performance on biomedical corpora. While DyVo [25] extended SPLADE's vocabulary to incorporate case-sensitive named entity terms, these named entity terms were used only as opaque expansion terms, meaning that their casing played no role in whether they were activated and the model's input remained uncased.

Recent evidence also suggests that cased models can improve dense retrieval for open-domain question answering. For example, ModernBERT [37] was compared with both base and large variants of uncased BERT (*BERT-base-uncased*, *BERT-large-uncased*), showing performance gains with certain dense retrieval techniques. Dense retrieval architectures allow relatively straightforward switching between cased and uncased models, as similarity is computed over dense vectors rather than token-level matches. In contrast, in LSR, where token-level matching is central, the impact of capitalization is more intricate and remains less studied.

More recently, researchers have begun extending LSR beyond encoder-only architectures, exploring encoder-decoder and decoder-only models [8,30,38,40]. These approaches generally rely on decoder backbones, which, unlike BERT, are typically released only in cased form. While this shift demonstrates that decoders can be effective for sparse retrieval, it also obscures the question of how much capitalization contributes to their performance, since uncased variants are not available for comparison. Consequently, the role of casing in decoder-based LSR remains underexplored.

In this paper, we focus on encoder-only models that provide both cased and uncased variants. This setting allows us to systematically investigate the effect of capitalization in LSR, including the design of pre-processing and post-processing strategies that might exploit cased inputs. By isolating these effects, we aim to clarify whether capitalization has a consistent effect on LSR; a question that remains open despite its potential significance for both encoder- and decoder-based retrieval.

3 Methodology

To evaluate the influence of casing and vocabulary space on model performance, we implemented a set of **pre-processing** and **post-processing** strategies, as illustrated in Fig. 1. These modifications were applied at different stages of training and evaluation to control the vocabulary space and avoid term frequency mismatches.

Pre-processing. We have introduced two techniques that are applied before passing the **input text** to the model:

1. **None**: The input text is used in its original form. For instance, case words such as "Apple" are preserved without modification.

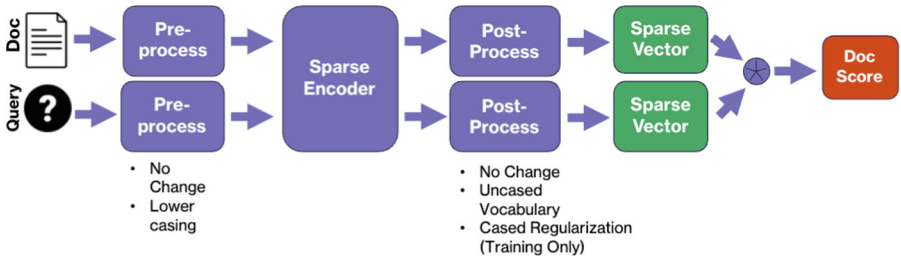

Fig. 1. Pipeline of cased models. Queries and documents first undergo a pre-processing step, followed by encoding, and then a post-processing step where sparse vectors are generated and compared. During post-processing, Cased Regularization is applied only during training as an additional loss.

2. **Lowercasing**: All input tokens are converted to lowercase. For example, "Apple" becomes "apple".

Post-processing. We have introduced three techniques that are applied to the **output embeddings** of the model:

1. **None**: The full output embedding is retained, preserving both cased and uncased token dimensions.
2. **Uncased Vocabulary Only**: Only the embedding dimensions corresponding to uncased tokens are preserved; those associated with cased tokens are zeroed. Words that exist exclusively in cased form are treated as their lowercase equivalents.
3. **Cased Regularizer (Training Only)**: During training, a regularization term is introduced to discourage the model from relying on cased token embeddings. This term penalizes the L2 norm of activations corresponding to cased dimensions. A polynomial warm-up schedule is used to ensure stable optimization. The regularization loss is defined as:

$$\mathcal{L}_{\text{Cased}} = w_t \sum_{j=1}^{N} (a^j_{\mathcal{I}_c})^2 = w_t \sum_{j=1}^{N} \sum_{i \in \mathcal{I}_c} \left(a^j_i\right)^2, \tag{3}$$

where w_t is the time-dependent regularization weight, N is the batch size, $\mathcal{I}_c$ is the set of indexes of the cased token dimensions, a^j_i is the i-th activation of the j-th sample, and $a^j_{\mathcal{I}_c}$ collects the activations over all cased dimensions for sample j.

4 Experimental Setup

Datasets. We trained and evaluated our models on the **MSMARCO Passage V1** [27] dataset, which contains approximately 8.8×10^6 passages. Additionally,

we assessed model performance on the **TREC Deep Learning 2019** (DL-2019) [3] and **TREC Deep Learning 2020** (DL-2020) benchmarks [2], both derived from the MSMARCO Passage V1 corpus and annotated with professionally judged relevance labels. To further test generalization, we evaluated our models on the **BEIR** [18] benchmark to assess out-of-distribution performance across diverse retrieval tasks.

Models. We evaluated the following models:

1. **BM25** [32]: A statistical retrieval method that operates on a bag-of-words representation and requires no training or encoder.
2. **SPLADE-BERT** [7]: A SPLADE model trained using BERT as the backbone. BERT is a deep bidirectional transformer encoder pre-trained with masked language modeling (MLM) and next sentence prediction (NSP). We fine-tuned publicly available BERT-base uncased[2] and cased[3] checkpoints.
3. **SPLADE-DistilBERT** [34]: A SPLADE model trained using DistilBERT as the backbone. DistilBERT is a lightweight variant of BERT, obtained through knowledge distillation, that is 40% smaller and 60% faster while retaining 97% of the original's performance. We used both the original uncased[4] and cased[5] versions.

Training. All models were trained using a **teacher-student distillation framework** with the **Margin-MSE loss**[6]. In this approach, each query is paired with a positive and a negative document, and the student model learns to match the *relative margin* provided by the teacher model. Formally, for a query q and a positive (d^+) and negative (d^-) document pair, the Margin-MSE loss is defined as:

$$\mathcal{L}_{\text{Margin-MSE}} = \frac{1}{B}\sum_{i=1}^{B}\left[f_S(q_i, d_i^+) - f_S(q_i, d_i^-) - f_T(q_i, d_i^+) + f_T(q_i, d_i^-)\right]^2 \quad (4)$$

where B is the batch size, f_S and f_T denote the student and teacher model scores respectively. This formulation encourages the student model to preserve the teacher's ranking of positive and negative documents.

To improve representation efficiency, we applied **FLOPs-based regularization** [28] to promote sparsity in the query and document embeddings as introduced in Equation (5):

[2] https://huggingface.co/google-bert/bert-base-uncased.
[3] https://huggingface.co/google-bert/bert-base-cased.
[4] https://huggingface.co/DistilBERT/DistilBERT-base-uncased.
[5] https://huggingface.co/DistilBERT/DistilBERT-base-cased.
[6] Teacher scores used to compute the pairwise margins are provided with https://www.dropbox.com/s/sl07yvse3rlowxg/vienna_triplets.tar.gz?dl=0.

$$\text{FLOPs} = \sum_{i=1}^{|V|} a_i^2 = \sum_{i=1}^{|V|} \left(\frac{1}{N} \sum_{j=1}^{N} a_i^j \right)^2 \tag{5}$$

where a_i represents the estimated activation probability of the i-the dimension across the batch.

Retrieval Evaluation. We assessed retrieval effectiveness using MRR@10 and R@1000 for the MSMARCO dataset, and nDCG@10 and R@1000 for all other datasets. Additionally, we measured FLOP loss on a subset of MSMARCO DEV queries to estimate efficiency, following the methodology used in SPLADE [10].

Implementation Details Our implementation is based on the publicly available SPLADE repository[7]. All hyperparameters follow those used in SPLADE-v2 [9]. For the introduced cased regularization term, we set the λ value to 0.2 for both queries and documents, using a scheduler to gradually increase it until 10,000 steps were reached, where it remained stable.

5 Results

We conduct experiments to evaluate the effect of casing on model performance using both in-domain datasets (MSMARCO, TREC DL-2019, TREC DL-2020, Table 1) and out-of-domain datasets (BEIR, Table 2). Across both BERT and DistilBERT, uncased models consistently outperform cased counterparts on in-domain tasks. This effect likely arises because MSMARCO-v1 queries are uncased and, as an open-domain corpus, MSMARCO-v1 provides limited utility for capitalization in disambiguation.

Pre-processing plays a crucial role in cased model performance. Without lowercasing, cased models exhibit a performance drop of about 5% relative to the uncased baseline ($\sim$ 36%). Applying lowercasing largely mitigates this gap, raising cased model accuracy to approximately 35.5%. For DistilBERT, the residual difference between the uncased variant and the cased variant with lowercasing as pre-processing is negligible ($\sim$ 0.30%). However, lowercasing does not improve efficiency, as DistilBERT with cased-lowering is less efficient than the uncased variant (4.46 vs. 2.95).

Post-processing strategies, such as constraining logits, reduce FLOPs by nearly 50% but slightly decrease accuracy ($\sim$ 0.20% MRR@10). While uncased vocabulary selection slightly improves accuracy compared to cased regularization (e.g., DistilBERT without pre-processing: 31.06% vs. 29.87% MRR@10), cased regularization has better efficiency (4.10 vs. 2.79 FLOPs). Both approaches, however, introduce additional complexity that limits their practical adoption.

For zero-shot transfer, we evaluate models on the BEIR benchmark (Table 2). Uncased models generally achieve stronger results (BERT-uncased leads on 9 of 14 datasets, and DistilBERT-uncased leads on 3). Nonetheless, cased models

[7] https://github.com/naver/splade/tree/main.

Table 1. Results on the MS MARCO Passage Collection (Dev, DL-2019, DL-2020). Models differ in pre-processing, post-processing, and query casing. Evaluation metrics are MRR@10, nDCG@10, and R@1000. Uncased models are trained on the original dataset with lowercasing only. Bold values indicate the best results.

Model	pre-processing	post-processing	MSMarco Dev		DL-2019		DL-2020		FLOPS
			MRR@10	R@1000	nDCG@10	R@1000	nDCG@10	R@1000	
BM25-RM3	Lowering	Uncased Voc. only	18.67	87.81	47.95	73.62	49.36	75.12	N/A
SPLADE BERT uncased	Lowering	Uncased Voc. only	35.77	**97.83**	**74.32**	**86.73**	**69.75**	82.08	3.16
SPLADE BERT cased	None	None	31.71	96.84	68.09	80.67	62.39	73.35	4.43
	None	Uncased Voc. only	30.89	96.73	66.44	80.72	61.00	73.17	3.59
	None	Cased Reg.	30.00	96.35	69.32	79.88	60.35	71.66	2.80
	Lowering	None	35.45	97.60	72.86	85.19	69.13	80.51	3.70
	Lowering	Uncased Voc. only	35.42	97.58	71.80	84.90	69.35	80.40	3.04
	Lowering	Cased Reg.	35.27	97.49	71.82	84.74	67.19	80.78	**1.99**
SPLADE DistilBERT uncased	Lowering	Uncased Voc. only	**35.91**	97.82	73.29	86.04	69.58	**82.78**	2.95
SPLADE DistilBERT cased	None	None	31.32	96.75	66.67	68.60	62.29	64.31	4.73
	None	Uncased Voc. only	31.06	96.78	65.42	81.09	62.78	73.68	4.10
	None	Cased Reg.	29.87	84.46	66.95	79.32	62.40	72.80	2.79
	Lowering	None	35.69	97.56	69.95	85.50	68.56	81.10	4.46
	Lowering	Uncased Voc. only	35.46	97.45	70.04	84.74	68.58	81.25	3.30
	Lowering	Cased Reg.	35.49	97.35	70.50	84.34	67.98	67.98	2.04

with lowercasing remain competitive, surpassing uncased models on NFCorpus (33.32% vs. 32.68%) and Quora (82.72% vs. 82.50%). Cased models without pre-processing generally underperform but still surpass BM25 across all datasets, with small gaps ($< 5\%$) relative to uncased models in Climate-FEVER, FEVER, and SciFact. Interestingly, BERT-cased without pre-processing outperforms both uncased and cased-lowering variants on Touché-2020, indicating that pre-processing choices influence retrieval effectiveness in specialized domains. Overall, uncased models remain the most reliable choice, though cased models with lowercasing prove effective in certain cases.

5.1 Answering the Research Questions

Having established the empirical results, we now return to our RQs and analyze them in light of the findings from both in-domain (MSMARCO, DL-2019, DL-2020) and out-of-domain (BEIR) evaluations.

Table 2. Results on the BEIR benchmark datasets. Models include BM25, BERT, and DistilBERT with cased and uncased variants. Cased variants are under different pre-processing conditions. The evaluation metric is nDCG@10. Bold values indicate the best results per dataset.

Corpus	BM25	SPLADE-BERT			SPLADE-DistilBERT		
	RM3	Uncased	Cased no pre-process	Cased lowering	Uncased	Cased no pre-process	Cased lowering
MS MARCO	47.95	**74.32**	68.09	72.86	73.29	66.67	69.95
ArguAna	34.24	50.44	47.96	49.72	**52.37**	46.59	50.81
Climate-FEVER	12.81	**23.18**	21.26	22.12	22.36	21.40	21.62
DBPedia	27.44	**42.95**	34.79	40.81	42.23	34.39	40.80
FEVER	42.73	**79.95**	75.95	77.37	77.98	76.70	76.87
FiQA-2018	25.26	**33.41**	30.47	32.66	32.06	30.04	32.33
HotpotQA	51.28	**68.74**	62.32	66.03	68.42	63.17	65.83
NFCorpus	32.22	32.68	30.53	**33.32**	32.97	30.07	32.66
NQ	23.09	**52.49**	48.24	51.12	51.48	47.77	50.54
Quora	76.76	82.50	81.60	**82.72**	81.91	80.75	81.65
SCIDOCS	14.71	15.46	12.34	14.85	**15.48**	11.96	15.23
SciFact	67.22	**69.16**	65.56	68.38	67.96	65.17	67.05
TREC-COVID	57.61	**71.38**	65.31	66.24	67.93	63.40	65.09
TouchÃľ-2020 (v1)	60.25	21.65	21.72	21.06	**67.93**	19.38	20.04
Avg. all	40.97	51.31	47.58	48.95	**53.88**	46.96	49.32
Avg. zero-shot	40.43	49.54	46.00	48.18	**52.39**	45.45	47.73
Best on dataset	0	**9**	0	2	3	0	0

RQ1: Casing Hinders In-Domain Retrieval Performance. Casing consistently hinders performance in in-domain retrieval. On MSMARCO Dev, cased BERT models without pre-processing underperform their uncased counterparts by approximately 4–5% MRR@10, with similar degradations observed in DL-2019 and DL-2020 with drops of 4–6% nDCG@10. This suggests that capitalization introduces spurious distinctions that are not beneficial in MSMARCO-style queries, which are generally lowercased. Importantly, this degradation persists across both BERT and DistilBERT, indicating that the effect is robust across encoder scales.

RQ2: Lowercasing Mitigates the Negative Impact of Casing. Lowercasing pre-processing substantially mitigates the negative impact of casing. For BERT-cased, lowercasing improves MRR@10 from 31.7 to 35.4 on MSMARCO Dev, reducing the gap to the uncased model to less than 0.3 points. A similar effect is observed for DistilBERT, where the difference between uncased and cased-lowercased models shrinks to ∼0.3 MRR@10. These results demonstrate that **cased models can be made competitive for LSR when paired with lowercasing**. Out-of-domain, cased-lowercased models occasion-

ally surpass uncased ones (e.g., NFCorpus: 33.3 vs. 32.7; Quora: 82.7 vs. 82.5 nDCG@10), highlighting that lowercasing not only closes the gap but can sometimes exploit residual benefits of cased pretraining.

RQ3: Post-processing Improves Efficiency, Not Accuracy. Post-processing techniques, such as restricting logits to uncased vocabularies or applying cased regularization, primarily affect efficiency rather than accuracy. For instance, on MSMARCO Dev, restricting logits reduces FLOPs by nearly 50% (4.1 → 2.8) with a marginal MRR@10 decrease (<0.2 points). However, improvements in efficiency come at the cost of added implementation complexity and vocabulary-specific design choices, limiting their practical adoption in new domains or pretrained variants without knowledge of the pre-trained vocabulary (e.g., in BioBERT [18]). Thus, while **post-processing is a viable efficiency optimization**, it is not an performance-enhancing strategy.

RQ4: Uncased Models Are More Robust for Zero-Shot Transfer. Uncased models are consistently more robust for zero-shot transfer. On BEIR, BERT-uncased achieves the highest nDCG@10 on 9 of 14 datasets, with a +3.3 point average margin over cased-no-pre-processing (49.5 vs. 46.0). DistilBERT-uncased similarly leads on 3 datasets, outperforming cased-no-pre-processing by +6.4 points on average. Though in selected datasets, Cased-lowercased models narrow this gap (e.g., BERT: 49.5 vs. 48.2), and they even outperform uncased models (e.g., NFCorpus, Quora). Nonetheless, cased-no-pre-processing models consistently underperform, barely exceeding BM25 on some datasets (e.g., Climate-FEVER, SciFact). Overall, **uncased models remain the most reliable for zero-shot transfer**, while cased-lowercased models provide a competitive alternative in certain domains, but never a consistently superior one.

5.2 Analysis of Cased and Uncased Tokens

To assess the role of casing in token generation, we analyzed query representations from the MS MARCO development set, where the majority of queries are lowercased. The goal of this analysis is to quantify the frequency with which input tokens are mapped to their cased or uncased forms in the output. Figure 2 presents the confusion matrices for BERT and DistilBERT under two pre-processing regimes: (i) no pre-processing and (ii) explicit lowercasing, with no post-processing being applied.

Across all models and conditions, uncased tokens account for the vast majority of output tokens, surpassing cased tokens by several orders of magnitude. This outcome is not surprising as lowercased words represent both the input data and the vocabulary of the models. The matrices also reveal systematic differences depending on whether pre-processing is applied. When no pre-processing is performed, cased input tokens are more frequently mapped to an uncased token rather than to a cased one. This indicates that even in the presence of cased input, both BERT and DistilBERT exhibit a preference for uncased vocabulary

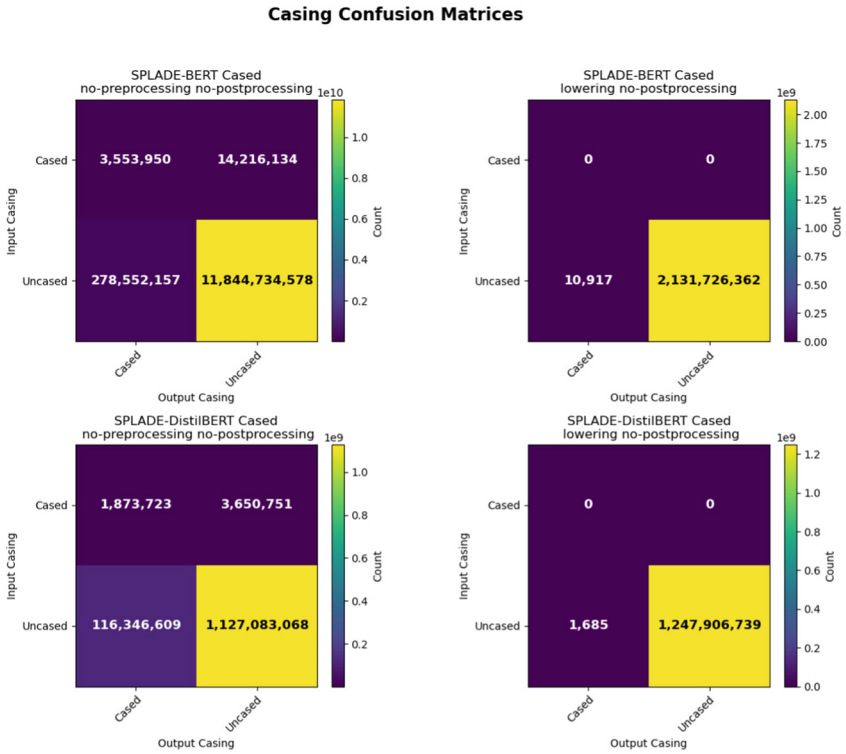

Fig. 2. Confusion matrices comparing input and output token casing across BERT and DistilBERT models under different pre-processing conditions. For both models, no post-processing method is used. Rows correspond to input token casing (cased vs. uncased), and columns represent the resulting output token casing. Values denote absolute token counts rather than normalized proportions, highlighting the relative dominance of uncased tokens across conditions.

items, likely due to the higher coverage and expressivity of uncased tokens in their vocabularies. Conversely, uncased inputs are overwhelmingly mapped back to uncased outputs, with only a very small proportion ($\approx 0.2\%$) producing cased outputs.

When input lowercasing is applied, the behavior changes dramatically. In this regime, both BERT and DistilBERT entirely limit the use of cased tokens to less than 1%, mapping all inputs to uncased outputs. This suggests that, despite being trained with cased vocabularies, the models effectively mimic the behavior of their uncased counterparts when forced into a lowercased input space. Such behavior likely reduces the risk of vocabulary mismatch, though it raises open questions about how the models internally represent and distribute capacity across cased and uncased subspaces.

Interestingly, in the case of BERT with lowercasing, only two specific tokens appear to be utilized in practice, namely `R` and `##R`. These tokens do not

correspond to meaningful English words, but rather seem to serve as anchor points within the cased subspace [21]. This observation warrants further investigation, as it could provide insights into how subword vocabularies adapt to pre-processing constraints and whether this impacts downstream retrieval effectiveness.

6 Conclusion

In this work, we systematically examined how the casing of backbone models affects LSR. Our results reveal that while cased models can underperform when used without pre-processing, applying lowercasing effectively recovers most of their retrieval effectiveness, bringing them on par with uncased models. Post-processing techniques, such as leveraging only uncased logits or regularizing cased tokens, offer notable efficiency improvements but only minor gains in accuracy, suggesting that their utility may be task- or resource-dependent. Importantly, these findings indicate that cased models remain viable for LSR, provided that pre-processing strategies are carefully applied. As future work, we want to extend this analysis to more recent cased architectures, like ModernBERT [37], which could unlock further improvements in both performance and efficiency.

Acknowledgments. This research was supported by the Hybrid Intelligence Center, a 10-year program funded by the Dutch Ministry of Education, Culture and Science through the Netherlands Organisation for Scientific Research, and by project VI.Vidi.223.166 of the NWO Talent Programme, which is (partly) financed by the Dutch Research Council (NWO). Additionally, this work benefited from a travel grant provided by the ELLIS Unit Amsterdam through the ELLIS Honors Program, which supported collaboration between the two universities.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Al-Hawari, F., Barham, H.: A machine learning based help desk system for IT service management. J. King Saud Univ. Comput. Inf. Sci. **33**(6), 702–718 (2021). https://doi.org/10.1016/J.JKSUCI.2019.04.001
2. Craswell, N., Mitra, B., Yilmaz, E., Campos, D.: Overview of the TREC 2020 deep learning track. In: Voorhees, E.M., Ellis, A. (eds.) Proceedings of the Twenty-Ninth Text REtrieval Conference, TREC 2020, Virtual Event [Gaithersburg, Maryland, USA], November 16-20, 2020. NIST Special Publication, vol. 1266. National Institute of Standards and Technology (NIST) (2020). https://trec.nist.gov/pubs/trec29/papers/OVERVIEW.DL.pdf
3. Craswell, N., Mitra, B., Yilmaz, E., Campos, D., Voorhees, E.M.: Overview of the TREC 2019 deep learning track. CoRR **abs/2003.07820** (2020). https://arxiv.org/abs/2003.07820

4. Crestani, F., Lalmas, M., van Rijsbergen, C.J., Campbell, I.: "is this document relevant? ... probably": a survey of probabilistic models in information retrieval. ACM Comput. Surv. **30**(4), 528–552 (1998). https://doi.org/10.1145/299917.299920
5. Dai, Z., Callan, J.: Context-aware sentence/passage term importance estimation for first stage retrieval. CoRR **abs/1910.10687** (2019). http://arxiv.org/abs/1910.10687
6. Dai, Z., Callan, J.: Deeper text understanding for IR with contextual neural language modeling. In: Piwowarski, B., Chevalier, M., Gaussier, É., Maarek, Y., Nie, J., Scholer, F. (eds.) Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, 2019, pp. 985–988. ACM, Paris, July 21-25 (2019). https://doi.org/10.1145/3331184.3331303
7. Devlin, J., Chang, M., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: Burstein, J., Doran, C., Solorio, T. (eds.) Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers), pp. 4171–4186. Association for Computational Linguistics (2019). https://doi.org/10.18653/V1/N19-1423
8. Doshi, M., Kumar, V., Murthy, R., P, V., Sen, J.: Mistral-SPLADE: LLMS for better learned sparse retrieval. CoRR **abs/2408.11119** (2024). https://doi.org/10.48550/ARXIV.2408.11119
9. Formal, T., Lassance, C., Piwowarski, B., Clinchant, S.: SPLADE v2: sparse lexical and expansion model for information retrieval. CoRR **abs/2109.10086** (2021). https://arxiv.org/abs/2109.10086
10. Formal, T., Piwowarski, B., Clinchant, S.: SPLADE: sparse lexical and expansion model for first stage ranking. In: Diaz, F., Shah, C., Suel, T., Castells, P., Jones, R., Sakai, T. (eds.) SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, July 11-15, 2021, pp. 2288–2292. ACM (2021). https://doi.org/10.1145/3404835.3463098
11. Guo, J., Fan, Y., Ai, Q., Croft, W.B.: A deep relevance matching model for ad-hoc retrieval. CoRR **abs/1711.08611** (2017). http://arxiv.org/abs/1711.08611
12. Hambarde, K.A., Proença, H.: Information retrieval: recent advances and beyond. CoRR **abs/2301.08801** (2023). https://doi.org/10.48550/ARXIV.2301.08801
13. Jones, K.S.: A statistical interpretation of term specificity and its application in retrieval. J. Document **60**(5), 493–502 (2004). https://doi.org/10.1108/00220410410560573
14. Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., Yih, W.: Dense passage retrieval for open-domain question answering. In: Webber, B., Cohn, T., He, Y., Liu, Y. (eds.) Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020, pp. 6769–6781. Association for Computational Linguistics (2020). https://doi.org/10.18653/V1/2020.EMNLP-MAIN.550
15. Khattab, O., Zaharia, M.: Colbert: efficient and effective passage search via contextualized late interaction over BERT. In: Huang, J.X., Chang, Y., Cheng, X., Kamps, J., Murdock, V., Wen, J., Liu, Y. (eds.) Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020. pp. 39–48. ACM (2020). https://doi.org/10.1145/3397271.3401075
16. Kunneman, F., Ferreira, T.C., van den Bosch, A., Krahmer, E.: Question similarity in community question answering: a systematic exploration of preprocessing

methods and models. In: Beuls, K., et al., (eds.) Proceedings of the 31st Benelux Conference on Artificial Intelligence (BNAIC 2019) and the 28th Belgian Dutch Conference on Machine Learning (Benelearn 2019), Brussels, November 6-8, 2019. CEUR Workshop Proceedings, vol. 2491. CEUR-WS.org (2019). https://ceur-ws.org/Vol-2491/abstract71.pdf

17. Laskar, M.T.R., Hoque, E., Huang, J.X.: Utilizing bidirectional encoder representations from transformers for answer selection. CoRR **abs/2011.07208** (2020). https://arxiv.org/abs/2011.07208
18. Lee, J., Yoon, W., Kim, S., Kim, D., Kim, S., So, C.H., Kang, J.: Biobert: a pre-trained biomedical language representation model for biomedical text mining. Bioinform. **36**(4), 1234–1240 (2020). https://doi.org/10.1093/BIOINFORMATICS/BTZ682
19. Lin, J.: A proposed conceptual framework for a representational approach to information retrieval. SIGIR Forum **55**(2), 4:1–4:29 (2021). https://doi.org/10.1145/3527546.3527552
20. MacAvaney, S., Nardini, F.M., Perego, R., Tonellotto, N., Goharian, N., Frieder, O.: Expansion via prediction of importance with contextualization. In: Huang, J.X., Chang, Y., Cheng, X., Kamps, J., Murdock, V., Wen, J., Liu, Y. (eds.) Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020, pp. 1573–1576. ACM (2020). https://doi.org/10.1145/3397271.3401262
21. Mackenzie, J., Trotman, A., Lin, J.: Wacky weights in learned sparse representations and the revenge of score-at-a-time query evaluation. CoRR **abs/2110.11540** (2021). https://arxiv.org/abs/2110.11540
22. Mallia, A., Khattab, O., Suel, T., Tonellotto, N.: Learning passage impacts for inverted indexes. In: Diaz, F., Shah, C., Suel, T., Castells, P., Jones, R., Sakai, T. (eds.) SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021, pp. 1723–1727. ACM (2021). https://doi.org/10.1145/3404835.3463030
23. Mallia, A., Siedlaczek, M., Mackenzie, J.M., Suel, T.: PISA: performant indexes and search for academia. In: Clancy, R., Ferro, N., Hauff, C., Lin, J., Sakai, T., Wu, Z.Z. (eds.) Proceedings of the Open-Source IR Replicability Challenge co-located with 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, OSIRRC@SIGIR 2019, Paris, July 25, 2019. CEUR Workshop Proceedings, vol. 2409, pp. 50–56. CEUR-WS.org (2019). https://ceur-ws.org/Vol-2409/docker08.pdf
24. Mitra, B., Craswell, N.: An introduction to neural information retrieval. Found. Trends Inf. Retr. **13**(1), 1–126 (2018). https://doi.org/10.1561/1500000061
25. Nguyen, T., Chatterjee, S., MacAvaney, S., Mackie, I., Dalton, J., Yates, A.: Dyvo: dynamic vocabularies for learned sparse retrieval with entities. In: Al-Onaizan, Y., Bansal, M., Chen, Y. (eds.) Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, November 12-16, 2024, pp. 767–783. Association for Computational Linguistics (2024). https://doi.org/10.18653/V1/2024.EMNLP-MAIN.45
26. Nguyen, T., MacAvaney, S., Yates, A.: Adapting learned sparse retrieval for long documents. In: Chen, H., Duh, W.E., Huang, H., Kato, M.P., Mothe, J., Poblete, B. (eds.) Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2023, Taipei, July 23-27, 2023, pp. 1781–1785. ACM (2023). https://doi.org/10.1145/3539618.3591943

27. Nguyen, T., Rosenberg, M., Song, X., Gao, J., Tiwary, S., Majumder, R., Deng, L.: MS MARCO: a human generated machine reading comprehension dataset. In: Besold, T.R., Bordes, A., d'Avila Garcez, A.S., Wayne, G. (eds.) Proceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic approaches 2016 co-located with the 30th Annual Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, December 9, 2016. CEUR Workshop Proceedings, vol. 1773. CEUR-WS.org (2016). https://ceur-ws.org/Vol-1773/CoCoNIPS_2016_paper9.pdf
28. Paria, B., Yeh, C., Yen, I.E., Xu, N., Ravikumar, P., Póczos, B.: Minimizing flops to learn efficient sparse representations. In: 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, April 26-30, 2020. OpenReview.net (2020). https://openreview.net/forum?id=SygpC6Ntvr
29. Porco, A., et al.: An alternative to FLOPS regularization to effectively productionize splade-doc. In: Ferro, N., Maistro, M., Pasi, G., Alonso, O., Trotman, A., Verberne, S. (eds.) Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2025, Padua, July 13-18, 2025, pp. 2789–2793. ACM (2025). https://doi.org/10.1145/3726302.3730163
30. Qiao, J., Nguyen, T., Kanoulas, E., Yates, A.: Leveraging decoder architectures for learned sparse retrieval. In: Wang, Z., Fang, J., Frisoni, G., Dai, Z., Meng, Z., Moro, G., Yilmaz, E. (eds.) Knowledge-Enhanced Information Retrieval - Second International Workshop, KEIR 2025, Lucca, Italy, April 10, 2025, Revised Selected Papers. Lecture Notes in Computer Science, vol. 16086, pp. 19–35. Springer (2025). https://doi.org/10.1007/978-3-032-02899-0_2,
31. Raffel, C., et al.: Exploring the limits of transfer learning with a unified text-to-text transformer. J. Mach. Learn. Res. **21**, 140:1–140:67 (2020). https://jmlr.org/papers/v21/20-074.html
32. Robertson, S.E., Walker, S.: Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In: Croft, W.B., van Rijsbergen, C.J. (eds.) Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval. Dublin, 3-6 July 1994 (Special Issue of the SIGIR Forum), pp. 232–241. ACM/Springer (1994). https://doi.org/10.1007/978-1-4471-2099-5_24
33. Robertson, S.E., Zaragoza, H.: The probabilistic relevance framework: BM25 and beyond. Found. Trends Inf. Retr. **3**(4), 333–389 (2009). https://doi.org/10.1561/1500000019
34. Sanh, V., Debut, L., Chaumond, J., Wolf, T.: Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. CoRR **abs/1910.01108** (2019). http://arxiv.org/abs/1910.01108
35. Schuster, M., Nakajima, K.: Japanese and Korean voice search. In: 2012 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2012, Kyoto, March 25-30, 2012, pp. 5149–5152. IEEE (2012). https://doi.org/10.1109/ICASSP.2012.6289079
36. Sennrich, R., Haddow, B., Birch, A.: Neural machine translation of rare words with subword units. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Volume 1: Long Papers. The Association for Computer Linguistics (2016). https://doi.org/10.18653/V1/P16-1162

37. Warner, B., et al.: Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. In: Che, W., Nabende, J., Shutova, E., Pilehvar, M.T. (eds.) Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, July 27 - August 1, 2025, pp. 2526–2547. Association for Computational Linguistics (2025). https://aclanthology.org/2025.acl-long.127/
38. Xu, Z., Feng, A., Tian, Y., Ding, H., Cheong, L.L.: CSPLADE: learned sparse retrieval with causal language models. CoRR **abs/2504.10816** (2025). https://doi.org/10.48550/ARXIV.2504.10816
39. Zamani, H., Dehghani, M., Croft, W.B., Learned-Miller, E.G., Kamps, J.: From neural re-ranking to neural ranking: Learning a sparse representation for inverted indexing. In: Cuzzocrea, A., et al., (eds.) Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, October 22-26, 2018, pp. 497–506. ACM (2018). https://doi.org/10.1145/3269206.3271800
40. Zeng, H., Killingback, J., Zamani, H.: Scaling sparse and dense retrieval in decoder-only LLMs. In: Ferro, N., Maistro, M., Pasi, G., Alonso, O., Trotman, A., Verberne, S. (eds.) Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2025, Padua, Italy, July 13-18, 2025, pp. 2679–2684. ACM (2025). https://doi.org/10.1145/3726302.3730225
41. Zhang, S., et al.: OPT: open pre-trained transformer language models. CoRR **abs/2205.01068** (2022). https://doi.org/10.48550/ARXIV.2205.01068
42. Zhao, T., Lu, X., Lee, K.: SPARTA: efficient open-domain question answering via sparse transformer matching retrieval. In: Toutanova, K., et al., (eds.) Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021, pp. 565–575. Association for Computational Linguistics (2021). https://doi.org/10.18653/V1/2021.NAACL-MAIN.47
43. Zhuang, S., Zuccon, G.: TILDE: term independent likelihood model for passage re-ranking. In: Diaz, F., Shah, C., Suel, T., Castells, P., Jones, R., Sakai, T. (eds.) SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, July 11-15, 2021, pp. 1483–1492. ACM (2021). https://doi.org/10.1145/3404835.3462922