Surface-Form Neural Sparse Retrieval: Robust Fuzzy Matching for Industrial Music Search

Paul Greyson
Amazon
San Francisco, CA, USA
pgreyson@amazon.com

Zhichao Geng
Amazon
Shanghai, China
zhichaog@amazon.com

Wei Zhang
Amazon
San Francisco, CA, USA
wizhang@amazon.com

Yang Yang
Amazon
Shanghai, China
yych@amazon.com

Abstract

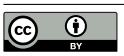
Music search at the scale of Amazon Music presents a unique challenge: queries frequently deviate from indexed metadata due to misspellings, transpositions, and phonetic variations, yet the retrieval system must operate under strict millisecond-level latency constraints. Our existing learning-to-retrieve system, the High Confidence Index (HCI), learns query-entity associations from customer behavior, relying on continual "exploration" to choose candidates. Traditional n-gram matching enables this exploration but suffers from poor semantic robustness and high noise, limiting the system's ability to learn from long-tail queries. In this work, we present a **robust neural sparse retrieval system** designed to maximize exploration efficiency. We adapt a state-of-the-art **inference-free** sparse retrieval architecture to the music domain, combining it with an effective **domain-specific granular subword tokenization strategy**. Our approach utilizes short-length token constraints (max 3 chars) to enforce the learning of surface-form robustness over lexical memorization. By pre-computing the neural embeddings and term expansions during the offline indexing phase, online processing is reduced to minimal tokenization and IDF weighting, achieving effectively zero latency overhead for query encoding. Evaluations on a 6M-document production corpus show an aggregate **91.4%** recall@10 (vs. **57.7%** for trigrams) at comparable throughput. Simulation of the HCI feedback loop demonstrates improved exploration efficiency, with **+0.8%** higher stabilized recall than production trigrams. Ablation studies indicate that our sparse training methodology drives the performance gains, while domain-specific pretraining provides a cost-effective alternative to large-scale general-purpose pretraining.

CCS Concepts

• **Information systems** → **Retrieval models and ranking**.

Keywords

Neural sparse retrieval, music search, fuzzy matching, surface-form robust modeling



SIGIR '26, Melbourne, VIC, Australia

ACM ISBN 979-8-4007-2599-9/2026/07
https://doi.org/10.1145/3805712.3808414

ACM Reference Format:
Paul Greyson, Zhichao Geng, Wei Zhang, and Yang Yang. 2026. Surface-Form Neural Sparse Retrieval: Robust Fuzzy Matching for Industrial Music Search. In *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '26), July 20–24, 2026, Melbourne, VIC, Australia.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3805712.3808414

1 Introduction

Music search queries frequently deviate from indexed metadata due to "query misspecifications" ranging from phonetic misspellings ("tayler" vs. "taylor") to character variations and transpositions [4, 20]. As illustrated in **Table 1**, these deviations present a dual challenge: the system must be robust enough to bridge significant lexical gaps yet efficient enough to meet strict millisecond-level latency budgets.

One method that Amazon Music uses to address relevance is the High Confidence Index (HCI), a learning-to-retrieve system that effectively memorizes query-entity affinities from historical engagement. However, HCI's learning depends critically on *exploration*—successfully matching a query to a candidate for the first time. Currently, exploration relies on traditional trigram matching. These methods are semantically brittle and technically constrained. For instance, OpenSearch's trigram configurations enforce minimum token lengths, discarding critical short terms such as "me". On the other hand, avoiding this loss by including whitespace tokens introduces order sensitivity that hinders matching transposed queries. As illustrated in Table 2, they often fail to distinguish incidental terms such as "songs" in "taylor swift songs" or break under structural variations like "p!nk", returning only exact matches or unrelated extensions. These limitations hinder the system's ability to efficiently explore and learn from long-tail queries.

Learned Sparse Retrieval (LSR) has emerged as a compelling solution to these challenges [5, 10]. By predicting sparse weight distributions over a vocabulary, LSR combines the contextual surface-form understanding of neural models with the efficiency of exact matching. Crucially, unlike dense retrieval [15] which requires specialized vector databases (ANN) [14, 19] and adds latency from online inference, LSR retains compatibility with standard **inverted indices** [2, 27]. This ensures high cost-effectiveness, seamless integration with existing production infrastructure, and inherent interpretability of matching features.

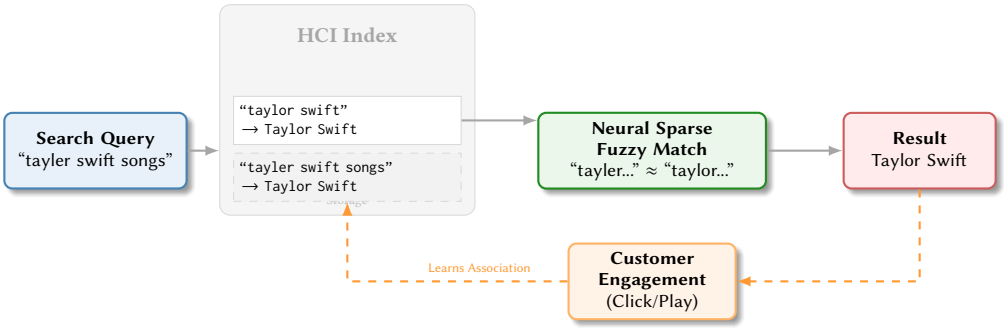


Figure 1: HCI exploration feedback loop. Neural sparse fuzzy matching (green) enables the system to bridge the gap between misspelled queries and canonical entities. Successful customer engagement (orange) creates a permanent record in the index (dashed box), converting future fuzzy matches into exact matches.

Table 1: Common cases of query misspecifications in music search.

Category	User Query	Target Entity
Simple Misspelling	tayler swift	Taylor Swift
Character Variation	p!nk	Pink
Transposition	carpenter sabrina	Sabrina Carpenter
Incidental Terms	taylor swift songs	Taylor Swift

Table 2: Comparison of neural sparse vs trigram matching on music queries. Scores shown in parentheses.

Query	Neural Sparse	Trigram
p!nk	p!nk (22.4), p nk (20.1), pnk (18.3)	p!nk (398), million dreams p!nk (28.9)
tayler swift	tayler swift (24.6), taylor swift (19.9), tyler swift (19.3)	tayler swift (458), tayler swift & post malone (89.1)

Building on this paradigm, we present a **robust neural sparse retrieval system** optimized for strict industrial latency. While standard LSR models (e.g., SPLADE [10]) often require costly online inference, we adopt an **inference-free asymmetric architecture** [8, 11, 26]. By pre-computing document representations via neural expansion during offline indexing, and using lightweight IDF weighting for online queries, we maintain the benefits of LSR with effectively zero latency overhead for query encoding.

To further address query misspecifications, we adopt a **data-driven granular tokenization strategy**. Instead of relying on rigid n-grams, our approach learns granular sub-word patterns [3, 17, 25] directly from search logs [13, 23], enabling the model to capture surface-form robustness. This allows the system to recognize the structural equivalence between variations without exact lexical matching. As demonstrated in the neural sparse column of Table 2, this strategy robustly handles complex variations like "p!nk" and "tayler swift" that typically fail with traditional methods.

Our main contributions are:

- The successful adaptation and deployment of an inference-free sparse retrieval architecture (SPLADE) to industrial music search, requiring no online neural encoding.
- An empirical validation of a domain-specific, granular tokenization strategy that outperforms traditional trigrams in robust fuzzy matching.
- Offline multilingual evaluations on a 6M-document production corpus showing **91.4%** aggregate recall@10 (vs. **57.7%** for trigrams) at comparable throughput, effectively achieving zero latency overhead for query encoding, alongside simulations demonstrating improved exploration efficiency.
- Ablation studies revealing that our sparse training pipeline is the primary performance driver, and that a modest domain-specific pretraining effectively replaces large-scale general-purpose pretraining.

2 System Overview

A key mechanism for fuzzy music search is the High Confidence Index (HCI), a probabilistic retrieval framework that ranks entities based on historical query-entity affinity. The system estimates the relevance score between a user query Q and an entity E through a factorization over previously observed queries Q':

$$\text{Score}(Q, E) \approx \sum_{Q' \in I} \underbrace{P(Q'|Q)}_{\text{Query Sim.}} \cdot \underbrace{P(E|Q')}_{\text{Behavioral}} \tag{1}$$

Here, $P(E|Q')$ represents the historical engagement probability stored in the index. Our work targets the **Query Similarity** term $P(Q'|Q)$. Although HCI effectively serves frequent queries via exact matching, it relies on an *exploration* mechanism to handle novel or misspecified queries (e.g., Q ="tayler", Q' ="taylor").

As illustrated in Figure 1, the system operates as a continuous feedback loop. Our **Neural Sparse Retrieval** module acts as the exploration engine, designed to robustly estimate $P(Q'|Q)$. To meet strict latency budgets, we use an **asymmetric inference-free architecture** [11]:

- **Offline Document Encoding:** In our retrieval framework, the "documents" to be indexed are historical queries (Q'). These are processed offline by our sparse encoding model,

where terms are expanded and sparse weights are predicted and stored in the OpenSearch inverted index.
- **Online Zero-Overhead Query Encoding:** At query time, we bypass neural inference. The user query Q is processed solely by our granular subword tokenizer and weighted via IDF.

Implementation-wise, this retriever operates as a pluggable module within the OpenSearch query plan. It replaces the legacy trigram matcher to provide a more robust similarity signal. When a user engages with a candidate retrieved via this signal, the system captures the event and updates the Behavioral term $P(E|Q')$, effectively converting a fuzzy exploration problem into a precise memory retrieval for future users.

3 Methodology

We adapt and optimize a neural sparse retrieval framework specifically for the lexical irregularities of music search. Our approach diverges from standard SPLADE LSR model [8–10, 18] by enforcing strict sub-lexical constraints and adopting a fully inference-free query path [11, 21, 26].

3.1 Surface-Form Robust Pre-training

Standard PLMs typically rely on fixed subword vocabularies (e.g., WordPiece), where minor perturbations can induce sharply different tokenizations and degrade robustness to severe query misspecifications in search [7, 24]. For example, a typo like "tayler" can map to a substantially different subword sequence than "taylor", reducing shared lexical evidence for the model to learn their equivalence. We address this via a joint tokenization and pre-training strategy tailored to music query noise. It is important to note that our design targets *robust lexical matching* rather than broad semantic retrieval. By constraining tokens to short lengths, we focus the model on learning surface-form equivalence (e.g., "ph" ≈ "f") rather than semantic synonyms, which minimizes the risk of irrelevant "hallucinations" common in dense retrieval.

Granular Tokenization. We train a custom SentencePiece Unigram model [16, 17] on 20 million music queries with a strict constraint: **maximum token length of 3 characters**. This constraint acts as an effective empirical heuristic, encouraging the tokenizer to decompose terms into granular subword constituents (e.g., character n-grams) rather than memorizing full words. Consequently, variations like "p!nk" and "pink" share significant sub-token overlap, providing a robust initialization for retrieval.

Domain-Specific MLM. Using this granular vocabulary ($|V| = 32,000$), we pre-train a BERT-base encoder from scratch on the query corpus. We employ the Masked Language Modeling (MLM) objective to teach the model the contextual semantics of these short sub-lexical units [6]. This produces a foundation model that is natively aligned with the structural irregularities of music search queries, unlike generic BERT models fine-tuned on standard text [1, 12].

3.2 Asymmetric Sparse Architecture

We adopt an inference-free architecture [8], utilizing a SPLADE-doc encoder to decouple neural computation from online retrieval.

Offline Document Encoding. Historical queries (Q'), treated as documents d, are encoded using the robust BERT model. SPLADE-doc repurposes the MLM head to project token embeddings $\mathbf{h}_t$ to the vocabulary space. The weight $w_j^{(d)}$ for token j is derived via max-pooling:

$$w_j^{(d)} = \max_{t \in d} \left(\log(1 + \text{ReLU}(\mathbf{E}_j^T \mathbf{h}_t + b_j)) \right) \tag{2}$$

This *neural expansion* assigns weights to unobserved but related tokens (e.g., misspellings). Optimized with InfoNCE [15] and FLOPS regularization [22], it yields a sparse, robust representation compatible with inverted indices.

Online Query Weighting. To ensure zero-latency inference, we bypass the neural encoder at query time. Query weights $w_j^{(q)}$ are derived solely from tokenization and corpus statistics [11]:

$$w_j^{(q)} = \mathbb{I}(v_j \in q) \cdot \text{IDF}(v_j) \tag{3}$$

This limits online computation to tokenization while retaining the expressiveness of offline expansions.

Indexing and Scoring. We implement dot-product retrieval $\sum w_j^{(q)} w_j^{(d)}$ via OpenSearch's `rank_features` field, which stores sparse weights as 16-bit floats for high-performance exact scoring.

Optimization Objective. We fine-tune the encoder using InfoNCE loss with hard and in-batch negatives, alongside FLOPS regularization to enforce sparsity:

$$\mathcal{L} = \mathcal{L}_{\text{InfoNCE}} + \lambda_{reg} \cdot \sum_{j \in \mathcal{V}} \left(\frac{1}{N} \sum_{i=1}^{N} w_j^{(d_i)} \right)^2 \tag{4}$$

where N is the batch size and d_i is the i-th document.

3.3 Weakly Supervised Data Construction

Since manual relevance labeling is infeasible at scale, we construct a weakly supervised dataset from aggregated behavioral logs.

Data Mining Strategy. We utilize 28 days of high-confidence query-entity playback logs. To train the model specifically for robust fuzzy matching (rather than broad semantic matching), we construct positive pairs (q, q^+) by aligning queries that: (1) Led to a playback of the *same* entity, (2) Have a length ratio $\text{len}(q_{\text{short}})/\text{len}(q_{\text{long}}) \geq 0.8$, and (3) Possess a character-level Levenshtein distance within a length-scaled threshold: $\max(1, \lfloor \text{len}(q)/10 \rfloor)$. This serves as an effective heuristic to capture common misspellings ("tayler swift" ↔ "taylor swift"), spacing differences ("sonideroaczino" ↔ "sonidero aczino"), transliterations ("radha kawach" ↔ "radha kavach"), and other lexical variations.

Hard Negatives. Generated through iterative mining with sparse retrieval. Epoch 1 uses sparse scores from the base BERT initialization via the MLM head; subsequent epochs use the previous sparse checkpoint for mining. We filter pairs sharing any playback entity to avoid false negatives.

Data Splitting. To prevent data leakage, we partition the dataset using connected components on the query-entity bipartite graph. This ensures that no query or entity in the test set has been seen during training, strictly evaluating the model's generalization capability. Initial empirical observations indicated that random splitting led to severe data leakage and artificially inflated performance,

making this graph-based partitioning a critical best practice for evaluation.

4 Experiments

4.1 Setup

Data & Implementation. We curate 1.19M documents and 237K test queries, utilizing graph-based partitioning to prevent leakage. Our best model is trained from scratch for ~16 hours on a p4d.24xlarge instance (8× A100 GPUs), running to 1,040,000 steps. We use an effective batch size of 120, a learning rate of 8.5×10^{-6} (2,000-step warmup), and progressive hard negative mining (4 negatives per query).

Metrics. We report Recall (R@k), Precision (P@k), and NDCG (N@k) for relevance. Efficiency is measured by query latency (ms) and index storage. Exploration capability is evaluated via simulation Recall against behavioral ground truth.

4.2 Offline Evaluation

Retrieval Performance. Table 3 compares our system against the production HCI Trigram baseline and fuzzy edit-distance on the production HC corpus (6.04M documents, 237K multilingual test queries). Our domain-trained neural sparse model reaches **91.4%** aggregate recall@10 compared to 57.7% for HCI Trigram and 83.4% for fuzzy edit-distance. All methods are measured on the same OpenSearch cluster (p4d.24xlarge, concurrency 32); neural sparse throughput is comparable to trigram matching (2,471 vs. 2,299 QPS) while producing substantially richer document representations. This confirms that surface-form robust modeling effectively bridges the lexical gaps in misspecified queries across diverse languages where rigid string matching cannot, while the inference-free query path preserves production throughput.

Table 3: Retrieval performance. Neural Sparse consistently outperforms the production HCI Trigram baseline on the production HC corpus (6.04M docs, 237K test queries, 8 languages). QPS on p4d.24xlarge, concurrency 32.

Method	N@1	R@10	QPS
Neural Sparse (ours)	**.557**	**.914**	2471
Fuzzy Edit-Distance	.508	.834	1465
HCI Trigram (prod.)	.475	.577	2299
Vanilla Trigram	.316	.521	2634

Efficiency & Cost. Latencies are minimized via the inference-free design: query encoding overhead is effectively zero (tokenization only), comparable to BM25. Sparse vectors add only ~25 tokens per document, increasing index size by a modest ~30%.

4.3 Ablation Studies

To isolate the contributions of our three design choices—domain pretraining, the sparse training pipeline, and the granular tokenizer—we evaluate three ablations on the same HC corpus (see Table 4).

A. Off-the-shelf sparse retriever. Evaluating an OpenSearch-released multilingual sparse model [11] without domain training achieves only R@10=0.795, underperforming fuzzy edit-distance (0.834). This confirms that general-purpose sparse models require domain adaptation for character-level lexical matching.

B. Domain data with a general-purpose backbone. Applying our full sparse training pipeline to a multilingual BERT backbone (without our domain pretraining) achieves R@10 between 0.906 and 0.937, depending on FLOPS regularization. This shows that our sparse training pipeline is the primary driver of quality, accounting for most of the gain over the off-the-shelf baseline.

C. Domain data with standard WordPiece tokenization. Substituting our granular SentencePiece tokenizer with a standard 24K WordPiece tokenizer (keeping domain pretraining and sparse pipeline) yields R@10=0.909 vs 0.949 for our SentencePiece tokenizer on a smaller evaluation corpus. By preventing canonical forms and misspellings from decomposing into entirely different sub-tokens, our 3-character ceiling constraint improves robustness to lexical perturbations.

Summary. The sparse training methodology is the primary performance driver. However, our domain BERT pretrained on just 20M queries matches or beats a multilingual BERT pretrained on billions of tokens. For teams with limited computational resources, this modest in-domain corpus provides a highly cost-effective substitute for large-scale general-purpose pretraining.

Table 4: Ablation summary on the HC corpus (6.04M docs). "Dims" denotes average non-zero dimensions per document.

Model	Tokenizer / Backbone	R@10	Dims	QPS
Ours (end-to-end)	SentencePiece-3 / Domain BERT	.914	86	2471
Domain WordPiece	WordPiece-24K / Domain BERT	.909†	—	—
BERT-ml (T1)	WordPiece-119K / BERT-ml	.906	37	3953
BERT-ml (T9)	WordPiece-119K / BERT-ml	**.937**	102	2026
Off-the-shelf AOS	WordPiece-105K / BERT-ml distill	.795	206	2152
Fuzzy Edit-Distance	—	.834	—	1465
HCI Trigram (prod.)	—	.577	—	2299

† Evaluated on a separate smaller corpus.

4.4 Production Log Replay

To assess the system's long-term evolution in a realistic setting, we conducted a replay backtest using production search logs. This evaluation simulates the continuous HCI feedback loop, where retrieved entities validated by user behavior are subsequently learned (indexed) for future retrieval.

We used 7 days of high-confidence behavioral logs (queries with >3 clicks) from production traffic. The simulation operates in discrete epochs, where each epoch represents one day of system operation. At epoch 0 (cold start), the index contains only catalog metadata with no behavioral data, and retrieval relies solely on standard text matching against entity fields (title, artist name, etc.). In subsequent epochs, we perform a partial index update: for each query in the evaluation set, we retrieve candidates using the current retrieval configuration, and successfully retrieved entities that match the behavioral ground truth are written back to the index as query-entity pairs with their engagement scores for use in the next epoch. We evaluate against ~750K queries from the 7-day log period where entities received customer clicks, representing validated relevance judgments from real user behavior.

We compare three retrieval strategies: (1) Cold Start using only text matching without behavioral data, (2) production baseline using exact match on historical queries plus trigram-based fuzzy matching, and (3) our approach using exact match plus Neural Sparse retrieval (limited to top-10 candidates to reduce noise from lower-confidence matches).

Table 5 shows that Neural Sparse significantly improves the discovery process. We ran the simulation for 15 epochs, with both methods converging by epoch 11. Starting from a cold-start recall of 0.8877, Neural Sparse converges to a higher plateau of 0.9627 versus 0.9548 for Trigram Fuzzy, recovering 7.5% more valid entities relative to the cold-start state and delivering an absolute gain of +0.79% over the production baseline. These results demonstrate the model's superior capability in surfacing long-tail candidates within the production environment, thereby effectively enhancing the overall HCI learning rate.

Table 5: Simulation results over 7 days of behavioral logs. Neural Sparse achieves better HCI coverage. Recall measured at k=25 to match production settings.

Configuration	Recall@25	vs. Cold Start
Cold start (no HCI)	0.8877	—
HCI Exact + Trigram Fuzzy	0.9548	+6.7%
HCI Exact + Neural Sparse	**0.9627**	+7.5%
Absolute improvement: +0.79% over Trigram Fuzzy		

5 Conclusion

We presented a neural sparse retrieval system for music search that significantly outperforms traditional n-gram matching on fuzzy query matching tasks. By training a custom granular tokenizer (max 3-char tokens) combined with a domain-specific BERT model on music search queries, we achieve an aggregate **91.4%** recall@10 at production-comparable throughput. End-to-end simulation shows consistently improved exploration efficiency over trigram matching. Our ablation studies demonstrate that the sparse training methodology is the primary driver of performance, and that a modest in-domain corpus is a cost-effective substitute for large-scale general-purpose pretraining. Our system demonstrates the viability of granular-tokenizer neural sparse methods for large-scale commercial search applications where latency and integration constraints are critical.

Acknowledgments

We thank the Amazon Music Search and Amazon OpenSearch teams for their support and infrastructure.

Speaker Biography

Paul Greyson is a Principal Engineer for search at Amazon Music where he currently leads development of machine learning based search solutions. He has worked in music technology for more than 25 years and received a bachelor's degree in Computer Science from UC Berkeley.

References

[1] Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A pretrained language model for scientific text. *arXiv preprint arXiv:1903.10676* (2019).
[2] Andrzej Białecki, Robert Muir, and Grant Ingersoll. 2012. Apache Lucene 4. In *SIGIR 2012 Workshop on Open Source Information Retrieval*.
[3] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics* 5 (2017), 135–146. doi:10.1162/tacl_a_00051
[4] Eric Brill and Robert C Moore. 2000. An improved error model for noisy channel spelling correction. In *Proceedings of the 38th annual meeting of the association for computational linguistics*. 286–293.
[5] Zhuyun Dai and Jamie Callan. 2020. Context-aware term weighting for first stage passage retrieval. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 1533–1536.
[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Jill Burstein, Christy Doran, and Thamar Solorio (Eds.). Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. doi:10.18653/v1/N19-1423
[7] Hicham El Boukkouri, Olivier Ferret, Thomas Lavergne, Hiroshi Noji, Pierre Zweigenbaum, and Junichi Tsujii. 2020. CharacterBERT: Reconciling ELMo and BERT for Word-Level Open-Vocabulary Representations From Characters. In *Proceedings of the 28th International Conference on Computational Linguistics*, Donia Scott, Nuria Bel, and Chengqing Zong (Eds.). International Committee on Computational Linguistics, Barcelona, Spain (Online), 6903–6915. doi:10.18653/v1/2020.coling-main.609
[8] Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE v2: Sparse Lexical and Expansion Model for Information Retrieval. doi:10.48550/ARXIV.2109.10086
[9] Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. 2022. From distillation to hard negative sampling: Making sparse neural ir models more effective. In *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval*. 2353–2359.
[10] Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE: Sparse lexical and expansion model for first stage ranking. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2288–2292.
[11] Zhichao Geng, Yiwen Wang, Dongyu Ru, and Yang Yang. 2024. Towards competitive search relevance for inference-free learned sparse retrievers. *arXiv preprint arXiv:2411.04403* (2024).
[12] Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. *arXiv preprint arXiv:2004.10964* (2020).
[13] Thorsten Joachims. 2002. Optimizing Search Engines Using Clickthrough Data. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 133–142. doi:10.1145/775047.775067
[14] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data* 7, 3 (2019), 535–547.
[15] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Online, 6769–6781.
[16] Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. *arXiv preprint arXiv:1804.10959* (2018).
[17] Taku Kudo and John Richardson. 2018. SentencePiece: A Simple and Language Independent Subword Tokenizer and Detokenizer for Neural Text Processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations (EMNLP Demo)*. 66–71. doi:10.18653/v1/D18-2012
[18] Carlos Lassance, Hervé Déjean, Thibault Formal, and Stéphane Clinchant. 2024. SPLADE-v3: New baselines for SPLADE. *arXiv preprint arXiv:2403.06789* (2024).
[19] Yu A Malkov and Dmitry A Yashunin. 2018. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence* 42, 4 (2018), 824–836.
[20] Bruno Martins and Mário J Silva. 2004. Spelling correction for search engine queries. In *International Conference on Natural Language Processing (in Spain)*. Springer, 372–383.
[21] Franco Maria Nardini, Thong Nguyen, Cosimo Rulli, Rossano Venturini, and Andrew Yates. 2025. Effective inference-free retrieval for learned sparse representations. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2936–2940.
[22] Biswajit Paria, Chih-Kuan Yeh, Ian EH Yen, Ning Xu, Pradeep Ravikumar, and Barnabás Póczos. 2020. Minimizing flops to learn efficient sparse representations. *arXiv preprint arXiv:2004.05665* (2020).

[23] Filip Radlinski and Thorsten Joachims. 2007. Active Exploration for Learning Rankings from Clickthrough Data. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 570–579. doi:10.1145/1281192.1281255

[24] Mike Schuster and Kaisuke Nakajima. 2012. Japanese and korean voice search. In *2012 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 5149–5152.

[25] Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*. 1715–1725. doi:10.18653/v1/P16-1162

[26] Xinjie Shen, Zhichao Geng, and Yang Yang. 2025. Exploring l0 Sparsification for Inference-free Sparse Retrievers. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2572–2576.

[27] Peilin Yang, Hui Fang, and Jimmy Lin. 2017. Anserini: Enabling the Use of Lucene for Information Retrieval Research. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 1253–1256. doi:10.1145/3077136.3080721