

Where Retrieval Quality Goes: Rethinking Stemming in Modern Search

Leo Galambos

Abstract—The effect of stemming is conditional on the lexical policy, ranker, query population, and evaluation objective; a result for one default dictionary does not measure the attainable performance of its adaptable engine. We examine these distinctions using English MIRACL development data: 799 queries over 32.9 million passages. At a common BM25 point, a task-adapted Radixor model improves its standard dictionary by 3.667% in nDCG@10, 4.043% in Recall@100, and 5.340% in AP@1000. All five recorded metric means improve over both standard Radixor and PyStemmer. The learned partition uses PyStemmer as its initializer and standard Radixor as its deployment substrate; positive assignments and negative boundary protection are both materialized in the model. An earlier, separately identified 164-evaluation study exposes scorer and objective sensitivity, including different best parameter coordinates with very small common-point regret. Fewer output terms but more term–document incidences exclude a pure-merge or pure-split account of the final representation change. We show why apparently conflicting historical findings need not estimate the same effect and distinguish this methodological explanation from an untested causal history of the field. The contribution is a controlled account of attainable policy improvement, retrieval depth, and evaluation scope. Reusing development judgments for selection and reporting limits the result to in-sample feasibility, not unseen-query superiority. A companion systems paper specifies the compilation contract and evaluates the adaptation mechanism against prior art.

Index Terms—stemming, information retrieval, BM25, lexical policy, model selection, MIRACL

I. INTRODUCTION

A standard stemming dictionary is one lexical policy, not a bound on what an adaptable stemming engine can achieve. This distinction matters when a benchmark result for a named stemmer is interpreted as a verdict on its underlying method. Our experiment supplies a concrete counterexample: at unchanged BM25 parameters, standard Radixor trails PyStemmer, while a task-adapted Radixor model exceeds it on all five recorded mean metrics. The same engine crosses that benchmark ordering through changes in lexical policy.

The adaptation has two semantic directions. Positive assignments prescribe which forms share retrieval evidence. Negative boundaries keep other forms out of selected classes even when a generic rule would admit them. Standard Radixor is reopened, extended, rebuilt, and paired with a compatible index; it is not replaced by another stemming engine. The search initializer is nevertheless the explicit PyStemmer partition, not the standard Radixor partition. This is a combined intervention using both a strong lexical prior and a deployable adaptable model, not an ablation of learning from Radixor alone.

The principal result is not an assertion of exclusive adaptability. Corpus-specific conflation, learned generalization, and per-query selection have established precedents [1]–[3]. Rather, the result demonstrates that *within-engine policy changes can alter the outcome attributed to the engine*. It provides a tractable experimental variable beyond a stem/no-stem switch or a single vocabulary-reduction measure.

A second contribution is interpretive. The phrase “stemming improves retrieval” suppresses the identity of the intervention and the quantity being measured. Different policies, scorers, query populations, cutoffs, and selection procedures need not yield the same sign. This is a reason to specify the effect, not to dismiss earlier experiments. Hull already argued for statistical and query-level examination rather than reliance on overall averages [4]. The present paper makes the conditional question explicit and links it to an executable policy change.

In a multi-stage search pipeline, the first stage also determines which passages a reranker can inspect; BERT passage reranking is one established example [5]. We evaluate the lexical stage, not a neural reranker or generated answers. Candidate recall and first-page ranking can favor different policies, so one aggregate score cannot describe all uses of stemming.

The paper addresses three connected questions: how scorer and objective choices condition policy comparisons; how much improvement is attained by the selected deployment over standard Radixor; and which conclusions survive the limits of same-development-set selection. The construction is shared with the companion *From Lexical Edits to Verified Stemming Policies* [6]. That paper gives the full positive/negative implementation, posting audit, and capability comparison. These are two views of the same experimental case, not independent replications.

II. BACKGROUND AND SCOPE

Porter supplies a widely used rule-based reference [7]. Paice evaluates linguistic under- and over-stemming against reference groups [8]; Flores and Moreira examine the relationship between stemming accuracy and retrieval [9]. Those intrinsic and extrinsic questions are related but not identical. A canonical lemma is not required as a retrieval label, and a morphologically plausible relation can be unhelpful for a particular ranked task.

Xu and Croft’s corpus-specific approach already adapts conflation classes using distributional evidence [1]. HPS combines unsupervised training and prediction of stems for unseen words [2]. Query-selective stemming can instead choose when to use a fixed analyzer [3]. Thus the historical landscape was not restricted to rigid language-wide policies. The detailed comparison in the companion paper distinguishes learning,

expressivity, contextual selection, and deployment verification rather than calling all alternatives equally configurable [6].

Component-interaction and retrieval-tuning studies motivate controlling the scorer [10], [11]. A shared operating point and a separately optimized configuration answer different questions; neither should be disguised as the other. The present final comparison uses one no-stemming-calibrated point, while the earlier Step 2 archive is analyzed as a distinct experiment. LLM-based stemming also explores richer contextual choices [12]; the policy studied here is deliberately scalar and context-free.

III. WHAT IS THE EFFECT OF STEMMING?

Let π be a lexical policy, β a scorer configuration, $\mathcal{C}$ a collection, $\mathcal{Q}$ a query population, and m an evaluation metric with a stated cutoff and judgment convention. The effect of replacing π_0 by π_1 is

$$\Delta_m(\pi_1, \pi_0 \mid \beta, \mathcal{C}, \mathcal{Q}) = \mathbb{E}_{q \sim \mathcal{Q}}[m_q(\pi_1, \beta) - m_q(\pi_0, \beta)]. \quad (1)$$

A sample mean estimates this particular contrast only under the corresponding sampling and selection assumptions. There is no policy-free quantity called *the* effect of stemming that all published systems necessarily estimate.

A. Three Questions That Must Not Be Collapsed

A default-policy comparison estimates $Q(\pi_a, \beta) - Q(\pi_b, \beta)$. A within-engine adaptation experiment estimates $Q(\pi_{a,\theta}, \beta) - Q(\pi_{a,0}, \beta)$ for a selected extension θ . A best-achievable comparison asks about $\sup_{\theta \in \Theta_a} Q(\pi_{a,\theta}, \beta)$ and the corresponding family for another engine. A single default run does not identify either supremum. Finding a better policy supplies a constructive lower bound on attainable performance, not a proof of global optimality.

If $Q(\pi_a, \beta) < Q(\pi_b, \beta)$ but $Q(\pi_{a,\theta}, \beta) > Q(\pi_b, \beta)$, the ordering of the two default artifacts was real, but its extension to the entire engine family was invalid. Our selected development case has this structure. It does not compare both families after equal-budget adaptation, so it also cannot establish their final relative ceilings.

B. Why Signs Can Differ Without a Contradiction

The mean difference is a weighted sum of per-query differences. For a two-stratum illustration, suppose an intervention changes a metric by $+0.10$ on one query type and -0.05 on another. If the first type has proportion p , the average is $0.15p - 0.05$, positive precisely when $p > 1/3$. These are illustrative values, not observations. The sign changes solely through the query mixture, without changing the intervention. Changing lexical boundaries or the metric can supply further sign changes. Hull’s query-level analysis and later selective stemming address precisely the danger of treating a mean as the whole phenomenon [3], [4].

Scorer calibration is another conditioning variable, not an automatic explanation for every negative result. A shared β holds ranking parameters constant; choosing a separate β_a^* for each method evaluates optimized system configurations. Neither design alone estimates the full interaction of policy

search and scorer search. Nor does a shift in the maximizer by itself show a large sensitivity: the Step 2 regret analysis below quantifies that distinction.

C. A Testable Modernization Claim

Radixor turns lexical boundaries into a model-level intervention while retaining its execution engine. This enables experiments on where additional conflation is useful and where it should be blocked, followed by verification of the policy actually indexed. It expands the controllable experimental design beyond a small menu of frozen defaults. The innovation supported here is that operational capability and its selected-task result, not the assertion that no previous system learned or configured stemming. The companion paper documents the older patch-command lineage and contrasts other forms of adaptability [6].

IV. STEMMING AS A VOCABULARY PARTITION

Let V contain the normalized surfaces occurring in the corpus or evaluated queries and let a scalar-output stemmer be a function

$$f : V \rightarrow S, \quad (2)$$

where S is an output-string space. The output labels themselves need not be linguistically meaningful. What matters for exact-term retrieval is the induced equivalence relation

$$w \sim_f v \iff f(w) = f(v). \quad (3)$$

The equivalence classes of $\sim_f$ form a partition Π_f of V . We call them *conflation classes*.

This formulation makes an important distinction. A stemmer can be globally “aggressive” in the sense of producing fewer classes, yet still place a locally harmful boundary; conversely a globally conservative stemmer can create one class that is too broad for a particular retrieval need. Therefore vocabulary reduction

$$|V| - |\Pi_f| \quad (4)$$

is only a coarse structural statistic. It does not identify which DF and TF masses are joined.

For the BM25 variant used here [13], the contribution of a query term t in document d is

$$\text{BM25}(t, d) = \text{IDF}(t) \frac{tf(t, d)(k_1 + 1)}{tf(t, d) + k_1 \left(1 - b + b \frac{|d|}{\overline{|d|}}\right)}, \quad (5)$$

with

$$\text{IDF}(t) = \log \left(1 + \frac{N - df(t) + 0.5}{df(t) + 0.5}\right). \quad (6)$$

Take $N > 0$, $\overline{|d|} > 0$, $k_1 > 0$, and $0 \leq b \leq 1$, with $0 \leq df(t) \leq N$, nonnegative original document lengths, and $0 \leq tf(t, d) \leq |d|$ under the scalar positional convention. Define a term contribution to be zero whenever its TF is zero; this avoids evaluating a zero-over-zero expression for an empty document at full length normalization. When two surface classes are merged, the induced retrieval term can gain documents (higher DF), gain repeated positions in existing documents (higher TF), or both. Higher DF reduces IDF, while additional matching

documents and TF can improve recall or ranking. Thus a conflation is not intrinsically good or bad for BM25: its effect depends on where the newly shared occurrences lie relative to the query and relevance distribution.

For a class C , $tf(C, d) = \sum_{w \in C} tf(w, d)$, whereas $df(C)$ counts the union of documents containing its members; DF is not the sum of surface DFs. Under this experiment’s one-output-per-position semantics, conflation leaves document length unchanged. A change in the best b would therefore concern the relationship between matching evidence and length, not documents becoming physically shorter.

An injective renaming of all class labels preserves these statistics and term matches. It preserves BM25 scores when query-position multiplicity, score aggregation, and document-length semantics are also fixed; exact ranked-list identity additionally requires the same numeric and tie-breaking behavior. The partition is thus the relevant abstraction under stated scorer semantics, not a claim about arbitrary retrieval implementations.

This motivates two policy primitives. A *positive assignment* specifies the retrieval output of a selected surface; assigning the same output to several surfaces requires them to share a class. A *conflation barrier* states that a surface must not enter a protected query class through generic stemming. The resulting deployment policy is conceptually

$$M_{\text{deploy}} = M_{\text{generic}} \oplus P^+ \oplus P^-, \quad (7)$$

where P^+ contains positive class assignments and P^- contains negative boundaries. The symbol $\oplus$ denotes ordered model extension, not arithmetic addition or an assumption that the effects are independent. These are semantic roles: a negative boundary is implemented by added patch commands, not by a negative frequency or a negative relevance label. The remainder of the paper constructs and tests this decomposition.

A. A Precise Retrieval-Equivalence Contract

The partition view is more than a metaphor. It identifies what a deployment must preserve and what it is free to change. Let $x_{d,i}$ be the normalized surface at original position i of document d , and let q_j be the surface at query position j . Define the positional support of class C by

$$\mathcal{P}_C = \{(d, i) : x_{d,i} \in C\}. \quad (8)$$

Then $tf(C, d)$ counts positions of d in $\mathcal{P}_C$, and $df(C)$ counts its distinct document identifiers.

Proposition 1 (label-invariant retrieval). Suppose two scalar analyzers f and g satisfy $g(w) = h(f(w))$ on every corpus and evaluated-query surface, where h is injective on the used output labels. Suppose they also preserve original positions, document lengths, query-position multiplicity, scoring arithmetic, and tie-breaking. They produce identical BM25 rankings after the label renaming.

Proof. For each used label s , injectivity gives $g(w) = h(s)$ exactly when $f(w) = s$. Every positional posting list therefore corresponds bijectively, so its TF, DF, and document support are unchanged. Each query position addresses its corresponding list. Every contribution to every document score is identical; fixed arithmetic and tie-breaking complete the argument. $\square$

Only classes reached by the evaluated queries require stored postings, provided their lists are constructed from *all* corpus occurrences and the full-corpus N and document lengths are retained. This is why a query-selective physical index can still evaluate the fixed query set exactly. It is not equivalent to searching a reduced document collection. A newly proposed query class during policy search requires its complete corpus support; a cache cannot silently treat an unmaterialized class as empty.

The proposition also explains why exact agreement with a human-readable lemma is a different property. Replacing every output label by an arbitrary unique identifier changes readable stems but not retrieval. A count of rewritten strings can change under this renaming even when the entire partition, index support, and ranking remain identical.

B. What Can Be Inferred from Aggregate Index Structure?

For the corpus census, restrict each analyzer to the set V_C of surfaces actually occurring at corpus positions and retain only nonempty fibers. Query-only surfaces belong to the wider retrieval contract but do not add census classes. Write $\Pi \preceq \Pi'$ for these corpus partitions when every class of Π is contained in a class of Π' . Define full-corpus term–document incidence as

$$I(\Pi) = \sum_{C \in \Pi} df(C) = \sum_d |\{C \in \Pi : tf(C, d) > 0\}|. \quad (9)$$

Proposition 2 (coarsening invariants). For scalar, one-output-per-position representations on the same observed corpus, $\Pi \preceq \Pi'$ implies

$$|\Pi'| \leq |\Pi|, \quad I(\Pi') \leq I(\Pi), \quad (10)$$

while the total collection frequency $\sum_C CF(C)$ remains constant.

Proof. Merging classes cannot increase their count. Within a document, replacing several present classes by their merged class cannot increase the number of distinct classes present. Summing over documents gives the second inequality. Each original position still belongs to exactly one class, so the total occurrence count is unchanged. $\square$

These conditions are necessary, not sufficient: matching aggregate counts do not identify a partition. Nevertheless, violations can reject a proposed pure-merge or pure-split account without access to every lexical pair. Section X-E applies this test to the supplied full-corpus census. It distinguishes a change in *boundary placement* from a movement along a single aggressiveness scale.

C. TF, DF, and Length: Three Distinct Mechanisms

Conflation changes the evidence supplied to BM25, not the definition of relevance. For two disjoint surface classes C_1, C_2 merged into C ,

$$tf(C, d) = tf(C_1, d) + tf(C_2, d), \quad (11)$$

$$df(C) = df(C_1) + df(C_2) - |D_{C_1} \cap D_{C_2}|, \quad (12)$$

where D_C is the set of documents containing C . The overlap term is essential: adding surface DFs would overcount documents. In contrast, collection frequencies add because the two original classes assign disjoint token positions.

For the score contribution above, put $f = tf(C, d)$, $k = k_1$, $A = 1 - b + b|d|/|\overline{d}|$, and $I_D = \log((N + 1)/(df(C) + 0.5))$. At fixed DF and length,

$$\frac{\partial \text{BM25}}{\partial f} = I_D \frac{(k + 1)kA}{(f + kA)^2} > 0 \quad (13)$$

for $k > 0$ and $A > 0$. At fixed TF and length, increasing DF reduces I_D . An extra class member can therefore increase the TF of one document while weakening the shared IDF of every document matching that query class. Neither effect alone determines the final ranking, which sums contributions from all query positions.

The frequently used explanation that a larger k_1 simply means “more precision” is too strong. Direct differentiation gives

$$\frac{\partial \text{BM25}}{\partial k} = I_D \frac{f(f - A)}{(f + kA)^2}. \quad (14)$$

Its sign depends on $f - A$. The parameter regulates saturation; it does not explicitly reward morphological correctness. Likewise,

$$\frac{\partial \text{BM25}}{\partial b} = -I_D \frac{(k + 1)fk(|d|/|\overline{d}| - 1)}{(f + kA)^2}. \quad (15)$$

Lower b weakens the relative penalty for above-average-length documents and the relative boost for below-average-length documents. The document lengths themselves remain unchanged by the scalar stemmers studied here. These derivatives explain mechanisms within the stated score formula [13]; they do not identify which mechanism caused a measured optimum to move.

A shift toward more selective conflation could, in principle, make repeated surviving matches more discriminative. Establishing that explanation would require measured class changes, TF/DF distributions, and relevance-conditioned document effects. The parameter shift alone does not establish increased under-stemming, reduced linguistic recall, or a causal transition toward precision. Throughout the paper, those explanations remain hypotheses unless supported by the specific intervention being discussed.

D. Where the Partition Abstraction Stops

The main construction uses a single output at every original position. Multiple alternatives require another object. Under position-sharing and maximum-alternative query scoring, a query position contributes

$$\max_{s \in A(q_j)} \text{BM25}(s, d), \quad (16)$$

not a sum that rewards a document repeatedly for alternative analyses of the same position. The overlap relation “shares at least one alternative” is generally not transitive, so it is not an equivalence relation on surfaces. The historical ADAPTIVE-ALL rows are retained as a separately named retrieval mode; they are not used in Propositions 1–2 or in the scalar sealing claim. The supplied final manifest documents the position/max

conventions of the benchmark, but it is not a separately pinned Step 2 executable environment.

Context-dependent analyzers also need a richer formulation because identical surface strings can acquire different outputs in different contexts. The present context-free formulation deliberately trades that expressiveness for a deterministic, auditable mapping. It does not claim that one equivalence class is a complete linguistic description of a homograph or a context-sensitive concept.

V. EXPERIMENTAL DESIGN

A. Collection and Retrieval Task

We use the English development split of MIRACL, a multi-lingual ad hoc retrieval dataset with native-speaker relevance judgments [14]. The experiment contains 799 English development queries and 32,893,221 English Wikipedia passages. The full indexed corpus contains 2,196,437,823 original tokenizer positions. The development split is used intentionally for both policy search and evaluation in this study. This is an oracle construction designed to measure feasibility and mechanism; it is not a held-out test protocol.

Each document is formed as title, a newline, and text. Text is normalized to NFC, case-folded, and tokenized into maximal non-empty sequences whose Unicode general category begins with Letter, Mark, or Number. No stopword removal and no accent folding are applied. Document length is the number of original tokenizer positions. For scalar-output methods, each position emits one term and TF is the number of distinct original positions carrying that term. Unjudged documents are treated as nonrelevant by the reported MIRACL-style metrics.

The persistent index materializes postings only for terms reachable from the query set, while the build simultaneously computes a full-corpus vocabulary census. This keeps the physical experiment tractable without approximating the postings needed by any measured query.

B. Methods

Four lexical representations are relevant to the final experiment:

- 1) NONE: normalized tokens are indexed unchanged;
- 2) PYSTEMMER: the English algorithm exposed by PyStemmer/Snowball;
- 3) RADIXOR-BASE: the default English Radixor primary output (`radixor-primary` in the supplied tables);
- 4) RADIXOR-SEALED: a compiled Radixor deployment model containing retrieval-specific positive conflations and conflation barriers, loaded with most-specific (last) lookup and the primary `stem_batch` API (`radixor-patched` in the supplied tables).

The manifest defines the effective scalar analyzer to return the original normalized token when the stemmer returns `None`; that fallback is part of f , not token deletion. The same effective analyzer must be used on corpus and query surfaces. The final sealed model is identified in that manifest by SHA-256

```
8f942746c8cf6c92fa71f3e0f3759698
4296f09884022db6fdbba74beda88a13e.
```

C. Two Experimental Stages, Not One Interchangeable Model

The available material contains two related but non-identical experiments. The earlier Step 2 archive tunes BM25 for a fixed adaptive dictionary created at 0.9/0.4. The later deployment experiment searches a PyStemmer-derived partition at the no-stemming-calibrated point and compiles a sealed Radixor model. Table I fixes these identities before any comparison. In particular, the label `radixor-primary` occurs in both archives but does *not* identify the same dictionary. Identical method names are insufficient provenance.

The numeric evidence has three different levels of reproducibility. Final benchmark summaries and the Step 2 response samples are machine-readable inputs from which tables and secondary calculations can be regenerated. Intermediate compiler/audit observations are a labelled transcription of the existing report; they cannot be independently replayed from those summaries. The propositions and secondary deductions in this revision are stated with assumptions and can be checked mathematically. Prospective experiments are identified as proposals, never inserted into the results as completed runs.

The reported no-stemming and PyStemmer common-point means agree across the two archives to their retained precision. This is a useful cross-file consistency check, not proof that all unrecorded environment details are identical. All final four-method comparisons use the final manifest and CSVs; all adaptive primary/all response claims use Step 2. No cross-stage difference is assigned to a specific dictionary edit without a matched intervention record.

D. Metrics and Objective

We report nDCG@10 for top-rank quality, Recall@100 and Recall@1000 for candidate coverage, AP@1000 for ranking quality through the stored run, and MRR@10 for the first relevant hit. These are distinct objectives, not interchangeable units of one quality scale. The scalar policy-search objective is

$$J = 0.5 \text{nDCG@10} + 0.4 \text{Recall@100} + 0.1 \text{AP@1000}. \quad (17)$$

The weights are a study design choice intended to retain strong top-rank quality while giving substantial weight to candidate-set recall. They are not proposed as a universal stemming objective.

Per-query contrasts are summarized with paired bootstrap intervals, two-sided sign-flip randomization statistics, and win/tie/loss counts. The manifest specifies 10,000 bootstrap replicates, 10,000 randomizations, and seed 20260914. We use the English language-level rows, not the separately resampled macro rows. Because these queries also selected the ranker and patched policy, these fixed-model resampling summaries do not estimate the generalization of the full selection procedure. Neither their nominal p values nor their nominal 95% intervals have established post-selection error-rate or coverage guarantees [15]. The archived Holm family covers four standard-Radixor contrasts, not sealed-model comparisons; it does not adjust the latter. Exact resampling, quantile, and tie-tolerance conventions require the missing evaluation implementation and per-query vectors.

E. BM25 Calibration

The initial experiments used $k_1 = 0.9$, $b = 0.4$. To avoid treating those convenient defaults as neutral, a separate tuner searches BM25 parameters using only the NONE representation and the objective in Eq. (17). The optimum observed for NONE is

$$k_1 = 1.125, \quad b = 0.3. \quad (18)$$

These values are then frozen for every lexical representation in the final deployment experiment. The separately identified historical Step 2 study retains its original per-method parameter searches.

This calibration matters empirically. Moving from 0.9/0.4 to 1.125/0.3 increases the objective from 0.496896 to 0.503734 for NONE, from 0.539477 to 0.545684 for PYSTEMMER, and from 0.521292 to 0.529014 for RADIXOR-BASE. Thus the observed gain from changing ranking parameters depends on the representation; this two-point comparison is not itself a formal interaction test or a measurement of every representation’s optimum. The calibration does not eliminate interaction or establish an unbiased universal setting. It establishes a common operating point selected on the no-stemming control, using the same development judgments subsequently used for policy search. In that sense it acts as a control against *default-parameter bias*: conclusions drawn at a convenient BM25 point can partly reflect the chosen operating region rather than the lexical policy alone.

The earlier “Step 2” archive concerns a different adaptive dictionary, identified as 2715a7dde336, and separate per-method BM25 tuning. Section VI analyzes that archive directly. It is not the source of the default-model row or evidence that the final sealed model has been jointly optimized. The present final-model checksum, methods, and frozen ranker are recorded in the accompanying manifest.

VI. HISTORICAL STAGE: THE SAME DICTIONARY, DIFFERENT RANKING OBJECTIVES

The earlier tuning archive contributes a distinct result that should not be collapsed into the sealed-model comparison. It concerns the adaptive dictionary 2715a7dde336, prepared under the anchor $\theta_0 = (0.9, 0.4)$, rather than the default model RADIXOR-BASE or the final sealed binary. We denote its scalar mode by ADAPTIVE-PRIMARY and its alternative-output mode by ADAPTIVE-ALL. This distinction resolves an apparent contradiction: the adaptive primary model can exceed PyStemmer on nDCG@10 while the *default* Radixor model does not.

A. Archive, Search Support, and Comparison Rules

The archive contains four evaluation CSVs, a best-per-method table, a common-parameter table, and a terminal tuning log. Each method has 41 distinct evaluated (k_1, b) points: 164 evaluations altogether, each with 799 queries and run depth 1000. The log records a coarse search around 0.9/0.4 followed by two refinements, with final step sizes 0.075 in k_1 and 0.05 in b . Across the sampled points, k_1 ranges from 0.3 to 1.5 and b from 0 to 0.8. The refinement centers differ between

TABLE I

EVIDENCE AND MODEL IDENTITIES. “AVAILABLE” REFERS TO THIS MANUSCRIPT PACKAGE, NOT TO EVERY ARTIFACT PRODUCED DURING THE ORIGINAL RESEARCH. MODEL IDENTIFIERS AND ORIGINAL SUMMARIES ARE PRESERVED; MISSING ARTIFACTS ARE NOT RECONSTRUCTED BY INFERENCE.

Stage	Lexical model / state	BM25 treatment	Available evidence and boundary
Step 2	Adaptive dictionary 2715a7dde336; primary and all modes	41 sampled points per method; own-best and common-point summaries	Seven original CSV/log files and README. Aggregate response data; no per-query Step 2 runs or dictionary bytes.
Final baselines	NONE, PYSTEMMER, default RADIXOR-BASE	Common 1.125/0.3	Manifest and final result summaries. Exact baseline software/model pins are incomplete.
Policy search and unsealed deployment	Explicit PyStemmer-derived oracle; positive-only compiled model	Frozen 1.125/0.3	Inherited intermediate report. Search checkpoint, audit inventories, and unsealed model are not supplied.
Sealed deployment	RADIXOR-SEALED, checksum beginning 8f942746c8cf	Frozen 1.125/0.3	Final means, paired summaries, index/work census, and manifest. Model checksum is present; model bytes are absent.

methods; the resulting samples are not one complete common rectangular grid.

We recompute every objective from the archived metric columns and verify the selected rows against both summary tables. All discrepancies are below 2×10^{-12} , consistent with the CSV precision. This is an arithmetic validation of the archived run, not a rerun of retrieval. No confidence interval for the location of a best point is available.

Two comparisons answer different questions. *Common-parameter evaluation* holds the scorer fixed and compares the supplied representations. *Per-method selection* compares each representation at its best observed point for the declared objective and search budget. Neither is universally preferable: the former isolates a controlled operating point, while the latter compares tuned systems. A paper must identify which comparison it is making rather than alternate silently between them.

B. Different Best Points, Small Local Regret

Table II reproduces the often-discussed parameter displacement: ADAPTIVE-PRIMARY selects 0.975/0.30 and ADAPTIVE-ALL selects 1.050/0.30, whereas NONE and PyStemmer select 1.125/0.30. Those different coordinates are real properties of the sampled archive. Their practical importance is better assessed together with the objective loss incurred by choosing a common point.

For method m , let Θ_m be its evaluated point set and define the *observed-grid regret*

$$\hat{r}_m(\theta) = \max_{\vartheta \in \Theta_m} J_m(\vartheta) - J_m(\theta), \quad \theta \in \Theta_m. \quad (19)$$

At the common point 1.125/0.30, regret is only 0.000200 for adaptive primary and 0.000316 for adaptive all. By contrast, changing the anchor 0.9/0.4 to the common point raises the objectives by approximately 0.006508 and 0.004817, respectively. Most of the available gain in this comparison comes from leaving the anchor region, not from resolving the small difference between the final nearby k_1 choices.

The shape is locally shallow. Adaptive primary at $k_1 = 1.050$ is only 0.000078 below its sampled best at 0.975. Figure 1 makes that scale visible instead of relying on an arrow between best-point coordinates. These observations motivate profiling a near-optimal region and a budget, rather than reporting one

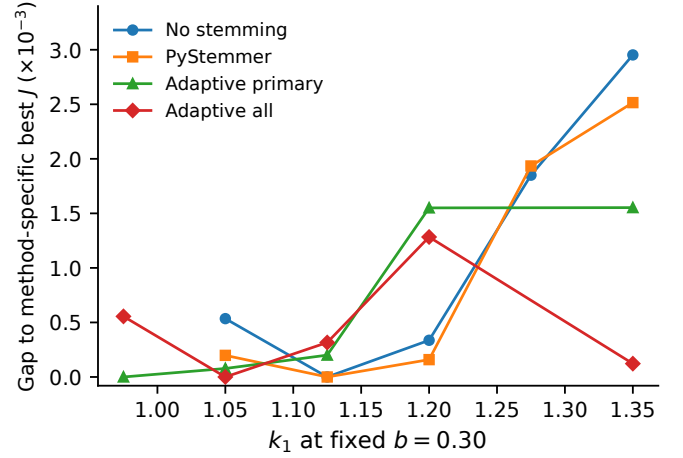


Fig. 1. Near-best Step 2 points on the measured $b = 0.30$ slice. The vertical axis is the gap to each method’s own best sampled objective, not its absolute effectiveness. Lines connect evaluated points only; unequal sampling and shallow gaps do not justify a smooth response surface or a unique population optimum.

parameter pair as though it were an invariant property of the language.

C. An Empirical Interaction, Not Yet a Causal Optimum Shift

The archive permits a direct descriptive check of non-additivity. If representation and ranker affected the objective additively, $J_m(\theta) = a_m + g(\theta)$, then the difference between two representations would be constant over shared parameter points. For adaptive primary and PyStemmer, 36 points are shared. Their objective difference ranges from -0.008915 to $+0.002085$ and is positive at three points, all with $b = 0$. At the anchor the difference is -0.007439 ; at the common calibrated point it is -0.007138 .

Thus an additive account cannot reproduce these observed means. The three favorable adaptive points do not make adaptive primary the best tuned system: for example, at 0.9/0 it has $J = 0.429300$, far below PyStemmer’s best observed 0.545684. The interaction is informative because it separates a *method comparison* from an *operating-point choice*, not because a reversal can be selected opportunistically.

TABLE II

HISTORICAL STEP 2: EACH METHOD AT ITS BEST OBSERVED J AMONG 41 EVALUATED POINTS. ADAPTIVE-PRIMARY AND ADAPTIVE-ALL USE DICTIONARY 2715A7DDE336; THESE ARE NOT THE DEFAULT OR SEALED-MODEL ROWS. NO GLOBAL OR STATISTICALLY UNIQUE OPTIMUM IS ASSERTED.

Method	k_1/b	nDCG@10	R@100	R@1000	AP@1000	MRR@10	J
NONE	1.125/0.30	0.339398	0.763944	0.884677	0.284580	0.393572	0.503734
PYSTEMMER	1.125/0.30	0.365427	0.829154	0.920418	0.313090	0.419755	0.545684
ADAPTIVE-PRIMARY	0.975/0.30	0.368561	0.808771	0.920202	0.309568	0.421073	0.538746
ADAPTIVE-ALL	1.050/0.30	0.365028	0.804117	0.918804	0.306849	0.421030	0.534846

TABLE III

STEP 2 OBJECTIVE AT THE ANCHOR, AT COMMON 1.125/0.30, AND AT THE METHOD-SPECIFIC BEST OBSERVED POINT. REGRET IS BEST MINUS COMMON, NOT AN ESTIMATE OF POPULATION LOSS.

Method	Anchor J	Common J	Best J	Regret
NONE	0.496896	0.503734	0.503734	0.000000
PYSTEMMER	0.539477	0.545684	0.545684	0.000000
ADAPTIVE-PRIMARY	0.532038	0.538546	0.538746	0.000200
ADAPTIVE-ALL	0.529713	0.534530	0.534846	0.000316

TABLE IV

PRE-ADAPTATION BASELINES AT THE COMMON BM25 POINT. VALUES ARE ROUNDED TO FOUR DECIMALS FOR READABILITY; THE FINAL TABLE GIVES ALL FIVE METRICS AT SIX DECIMALS.

Method	nDCG@10	R@100	AP@1000	J
NONE	0.3394	0.7639	0.2846	0.5037
PYSTEMMER	0.3654	0.8292	0.3131	0.5457
RADIXOR-BASE	0.3575	0.8005	0.3007	0.5290

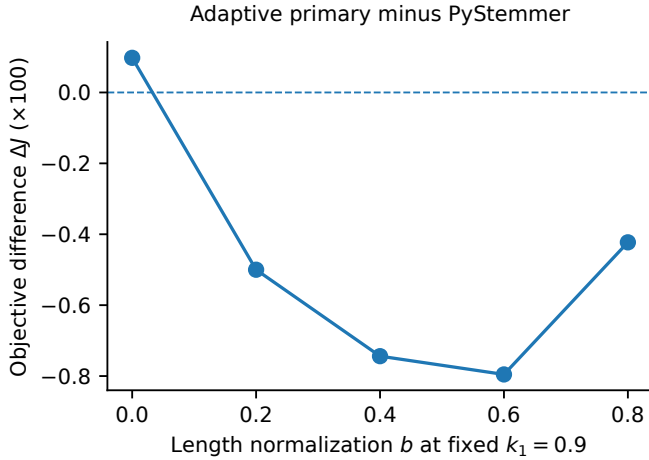


Fig. 2. A measured component interaction: adaptive primary minus PyStemmer at shared $k_1 = 0.9$ points. The sign changes at $b = 0$, but both systems are substantially weaker there than near their best points. This is not a recommendation to select a poor ranker to obtain a favorable method ordering.

A different, stronger claim would be that optimizing the dictionary *caused the ranker's optimum to move*. To identify that shift, the initial and learned dictionaries must both be profiled under a matched protocol, including common search support and an uncertainty analysis. The training anchor is not evidence that the initial dictionary's optimum was 0.9/0.4. Step 2 establishes that the learned dictionary's best sampled point differs from its training anchor and that method contrasts depend on BM25. It does not, by itself, identify a before/after causal displacement of the optimum. This distinction preserves the research question without overstating its present answer.

D. Which Quality Was Favored?

At their own J -selected points, adaptive primary exceeds PyStemmer on nDCG@10 by 0.003135 and on MRR@10 by

0.001318, but trails on Recall@100 by 0.020383 and AP@1000 by 0.003521. Recall@1000 is almost equal, with a difference of -0.000216 . The common-point comparison shows the same top-rank versus Recall@100 ordering. This is the empirical motivation for asking *where* the retrieval difference occurs, rather than declaring either method uniformly better.

The archive also shows that the best scorer depends on the metric even *without changing the dictionary*. For adaptive primary, the best observed nDCG@10 occurs at 0.975/0.30, Recall@100 at 1.050/0.30, Recall@1000 at 1.200/0.40, and AP@1000 at 1.125/0.30. PyStemmer's best nDCG@10 is at 1.050/0.30, not at its J -selected 1.125/0.30. Appendix C records all selected locations. These are maxima over the original J -driven samples, not independently budgeted searches for each metric.

The Step 2 results therefore establish an observed compromise, not a necessary frontier. They neither prove that more under-stemming caused the top-rank advantage nor show that a better policy could not improve both quantities. The later oracle-and-sealing construction addresses precisely that remaining possibility at a frozen ranker.

VII. BASELINES AT THE CALIBRATED OPERATING POINT

Table IV reports the three pre-adaptation methods at $k_1 = 1.125$, $b = 0.3$. Both stemmers substantially improve over no stemming, but PyStemmer retains a clear recall advantage over the default Radixor model.

The default Radixor shortfall is not uniform over queries. For Recall@100, default Radixor and PyStemmer tie on 729 of 799 queries, while Radixor is higher on 14 and lower on 56. At Recall@1000, they tie on 766 queries. The mean gap is therefore concentrated rather than universal. This concentration is one reason to investigate individual conflation boundaries instead of attributing the result to a global property such as “light” versus “heavy” stemming.

VIII. FROM PYSTEMMER TO AN EXPLICIT RETRIEVAL ORACLE

A. Full Projection: No Hidden BM25 Mechanism

Before editing any partition, we first ask whether PyStemmer’s observed retrieval result is fully explained by its lexical equivalence classes. For every query-reachable PyStemmer stem, we collect all corpus surface forms that PyStemmer maps to that stem and build an explicit projected partition. The inherited experiment report states that scoring this projection with the same BM25 semantics exactly reproduced the normal PyStemmer benchmark at the calibrated operating point, with the same check for default Radixor. The present aggregate package does not independently replay that equality.

This is an implementation consistency check: within the reported projection experiment, running the analyzer and explicitly indexing its induced classes give the same measured retrieval behavior. At fixed token positions and BM25 semantics, the retrieval behavior is explained by the partition: joining forms changes the postings, TF, DF, and therefore ranking.

A preceding bounded adaptive candidate generator already covered 99.792% of PyStemmer projection surfaces, 99.9998% of their collection frequency, and 99.9997% of their document-frequency mass. This establishes high coverage by aggregate mass, but does not exclude an important contribution from a small, query-sensitive missing tail. A full projection makes the equivalence exact under Proposition 1; the reported implementation check is restricted to the query-reachable space used here.

B. Oracle Search Space

The oracle search starts from the complete PyStemmer query-reachable partition. Candidate relationships come from the PyStemmer classes, Radixor classes, and lexical candidate generation. A selected undirected edge means that its endpoints belong to one connected conflation component. The search supports three exact local operations:

- ADD: join a candidate edge not currently selected;
- REMOVE: delete a selected edge when doing so improves the objective;
- SWAP: remove one selected edge and add another.

The search evaluates changes directly against the MIRACL development judgments under the frozen BM25 parameters. The inherited search description uses shortlists and caches for feasibility and exact explicit-projection scoring of accepted operations. The supplied aggregates do not verify that executable or its complete acceptance trace.

This procedure should be read as an *oracle* for two reasons. First, relevance judgments from the same development queries guide the partition search. Second, the objective in Eq. (17) is directly optimized. The best observed policy is an attained in-sample result within the explored search space. Bounded shortlists do not certify an exhaustive local optimum, much less a global ceiling. The term *oracle* denotes access to the evaluation judgments, not an optimality guarantee.

C. What the Oracle Changes

The search quickly finds that some fixed PyStemmer conflations are harmful for the chosen objective. Early accepted operations are predominantly removals, indicating that the improvement is not achieved simply by “more stemming.” Instead, the oracle retains broad useful conflations while separating selected surfaces whose shared postings reduce ranking utility.

After 51 accepted operations, the selected graph contains approximately 10,780 edges and the objective is near 0.550. This is already above the PyStemmer starting point of 0.545684. More important than the absolute number is the direction of change: the oracle demonstrates that the fixed PyStemmer partition is not locally optimal for this collection/objective, even though it is a strong baseline.

D. Partition Search Is Not Edge Search

The selected graph is a representation of an equivalence relation; its edges are not independent retrieval decisions. Adding an edge inside one connected component changes no class and therefore cannot change retrieval under the partition model. Removing an edge from a cycle also leaves the partition unchanged. A removal can split a class only when connectivity changes. Conversely, one edge joining large components can move many surfaces and occurrences at once. The reported edge count is therefore not a count of independent linguistic rules or an estimate of the number of effective degrees of freedom.

This distinction suggests concrete implementation checks. Candidate states should be canonicalized by their induced partition, or at least detected when an edge edit is a partition no-op. Cached scores must be keyed by all state that affects scoring: class memberships, complete corpus postings, query-class assignments, ranker parameters, and scoring conventions. A cache keyed only by edge identity can become invalid after neighboring classes change. These are requirements of an exact implementation, not assertions that every optimization was present in the unavailable optimizer revision.

The influence of an edit can also be nonlocal at query level. A class’s changed DF affects every document contribution for that query term, including documents whose own TF did not change. Reusing scores only for newly added documents would miss that effect. If a query contains two original positions that now map to one class, its position multiplicity must follow the declared scorer rather than being silently deduplicated. Periodic full rescoring of an incumbent is therefore an important correctness check on incremental search, not merely an engineering convenience.

IX. FROM THE SELECTED PARTITION TO THE DEPLOYED MODEL

The selected policy is installed as explicit positive assignments in the reopened standard model. The inherited report verifies 54,745 assignments, but the first compiled realization reaches only approximately $J = 0.545322$, below the explicit search trajectory near 0.550. Correct assigned surfaces do not

exclude unassigned surfaces from reaching the same outputs through generalization.

The reported audit finds 11,291 unintended surfaces entering protected query classes. Negative boundary extensions then exclude those memberships, with scalar lookup retaining the intended primary output. The first post-repair scan reports no remaining outsiders over its 6,046,942-surface inventory and preservation of all 1,684 protected query classes. The inventory differs by 16 from the final no-stemming census; closure is restricted to the actual scanned set. The companion paper derives the positional accounting identity and describes the shared patch-command guard, lookup precedence, and serialization checks [6].

This separation between policy selection and policy realization is essential. A failed compiled realization is not proof that the selected relation was useless; equally, matching a scalar objective is not proof that compilation preserved the relation. The final summaries below describe the deployed result, while missing intermediate artifacts limit independent reconstruction of the entire causal path.

X. FULL-CORPUS RETRIEVAL RESULTS

A. Aggregate Effectiveness

The sealed model is indexed over the complete 32.9-million-passage English MIRACL corpus and evaluated with the same frozen BM25 parameters used by every method. Table V reports the final result.

Relative to PyStemmer, the sealed objective increases by 0.004435 after rounding, or 0.813%. Relative to the default Radixor model, the increase is 0.021105 after rounding, or 3.989%. The transition from the unsealed objective near 0.545322 to 0.550119 recovers about 0.004797, closely matching the deficit that motivated the leakage audit.

This before/after sequence is more informative than the final score alone:

$$\begin{aligned} J_{\text{default Radixor}} &= 0.529014, \\ J_{\text{positive patch}} &\approx 0.545322, \\ J_{\text{sealed patch}} &= 0.550119. \end{aligned} \quad (20)$$

The positive-assignment stage raises the reported objective toward the PyStemmer reference. The negative stage addresses the audited mismatch between intended and realized memberships. This is a staged combined intervention, not a factorial estimate of two independent effects: no negative-only model is available, and the positive-only intermediate is retained as a reported approximation.

The arithmetic in Fig. 3 supports a before/after account of sealing. It does not establish exact equality to the final explicit oracle: the terminal oracle checkpoint and per-query runs needed for that comparison are not included in the current package.

B. How Much Better Is the Adapted Standard Dictionary?

The most direct capacity test is not a comparison between unrelated engine names. It is the comparison between the standard Radixor dictionary and its adapted deployment. Table VI exposes that improvement rather than requiring the

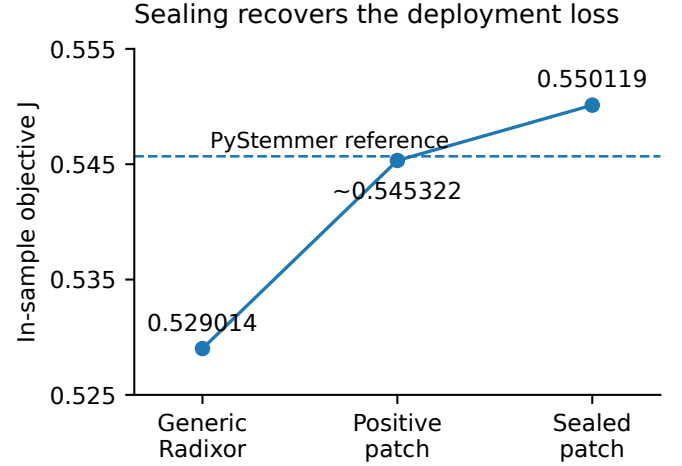


Fig. 3. Deployment-stage comparison at fixed BM25. The approximate unsealed point is preserved from the inherited intermediate report; the base, reference, and sealed points use final CSV means. Connecting segments indicate the stages compared, not a recorded optimizer trajectory. The truncated vertical scale resolves the small changes.

reader to subtract rows of Table V. Tokenization, collection, scalar output semantics, and BM25 parameters are shared; the policy artifact and its declared lookup configuration implement the intervention.

The nDCG@10 and Recall@100 increases are approximately 3.7% and 4.0%, respectively; AP@1000 increases by about 5.3%. The all-five-metric pattern matters as much as the 3.989% composite gain: the improvement over the standard model is not obtained by exchanging a lower reported recall mean for a higher top-rank mean. Nor does the original standard model’s deficit relative to PyStemmer establish an architectural deficit that survives adaptation. With the positive and negative extension, the observed mean ordering reverses at the same scorer.

A precise algebraic consequence is useful. For either fixed baseline b in {standard Radixor, PyStemmer}, all five coordinate differences $\bar{m}_j^{\text{adapted}} - \bar{m}_j^b$ are positive. Hence

$$\sum_j \lambda_j (\bar{m}_j^{\text{adapted}} - \bar{m}_j^b) > 0 \quad (21)$$

for any $\lambda_j \geq 0$ with at least one positive weight over these five observed metrics. The selected final comparison is therefore not an artifact of one particular nonnegative weighting of its reported means. This does not describe retraining with another objective, certify statistical Pareto dominance, or guarantee improvement for each query. It is a statement about the already-selected runs.

The evidence supports a realizable combined adaptation of the standard dictionary. It does not separately estimate the causal contribution of the search initializer, each positive assignment, each barrier, or the lookup configuration; those require additional matched ablations. Predictive validation must likewise freeze the complete adapted policy before using held-out relevance judgments.

For the sealed-minus-PyStemmer objective difference, the weighted nDCG@10, Recall@100 and AP@1000 terms con-

TABLE V

FINAL FULL-CORPUS EFFECTIVENESS AT $k_1 = 1.125$, $b = 0.3$. THE SEALED POLICY WAS SELECTED ON THESE SAME 799 DEVELOPMENT JUDGMENTS. BOLD ENTRIES IDENTIFY THE LARGEST OBSERVED MEANS, NOT HELD-OUT SUPERIORITY. J IS RECOMPUTED FROM UNROUNDED MEANS IN DATA/QUERY-SUMMARY.CSV.

Method	nDCG@10	Recall@100	Recall@1000	AP@1000	MRR@10	J
NONE	0.339398	0.763944	0.884677	0.284580	0.393572	0.503734
RADIXOR-BASE	0.357492	0.800485	0.912621	0.300742	0.410104	0.529014
PYSTEMMER	0.365427	0.829154	0.920418	0.313090	0.419755	0.545684
RADIXOR-SEALED	0.370600	0.832846	0.921878	0.316801	0.424842	0.550119

TABLE VI

STANDARD RADIXOR TO ADAPTED RADIXOR AT FIXED BM25, ON THE SAME 799 DEVELOPMENT QUERIES USED IN SELECTION. RELATIVE GAINS USE UNROUNDED MEANS. THESE ARE OBSERVED EFFECT SIZES, NOT HELD-OUT ESTIMATES OR SIGNIFICANCE CLAIMS.

Metric	Standard	Adapted	Gain (%)
nDCG@10	0.357492	0.370600	+3.667
Recall@100	0.800485	0.832846	+4.043
Recall@1000	0.912621	0.921878	+1.014
AP@1000	0.300742	0.316801	+5.340
MRR@10	0.410104	0.424842	+3.594
J	0.529014	0.550119	+3.989

tribute 0.0025867913, 0.0014768461 and 0.0003711489, respectively: approximately 58.3%, 33.3% and 8.4% of the defined score change. This is an additive decomposition of the chosen utility, not a causal attribution to separate retrieval mechanisms; the same rank changes can affect several metrics.

C. Paired Query Diagnostics

Table VII reports descriptive paired contrasts between the sealed model and PyStemmer. Because the sealed policy was optimized on the same queries, these p -values must not be interpreted as confirmatory evidence for unseen data. Concentration is assessed from effect sizes and win/tie/loss counts, not inferred from a small nominal p value.

The nDCG contrast is especially concentrated: 741 of 799 queries tie to numerical tolerance, with 49 sealed-model wins and 9 losses. Recall@1000 differs on only two queries, both in favor of the sealed model. This concentration echoes Hull’s warning that aggregate stemming results should be inspected at query level [4].

Figures 4 and 5 separate effect size from its support across queries. In particular, a positive mean Recall@1000 difference rests on only two non-tied queries. No per-document rank-transition analysis is available here, so the aggregate metrics do not identify which relevant passages moved between depth bands.

Against the default Radixor model, the sealed policy improves all reported means: nDCG@10 by 0.013109, Recall@100 by 0.032361, Recall@1000 by 0.009257, AP@1000 by 0.016059, and MRR@10 by 0.014737. These contrasts quantify how much retrieval-relevant policy remains available beyond the generic lexical model on this collection.

Sealed Radixor minus PyStemmer

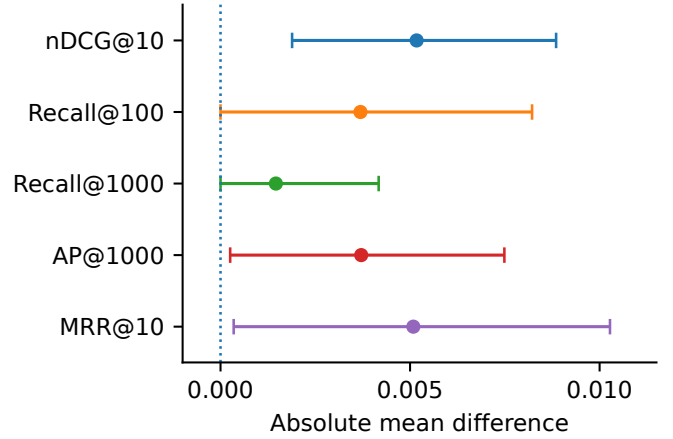


Fig. 4. Where the measured differences appear. Points are absolute differences on each metric’s 0–1 scale; bars are the reported descriptive bootstrap intervals. The rows concern different quantities and must not be joined into a smooth retrieval-depth curve or interpreted as selection-adjusted significance.

D. Vocabulary Size and Query Work

If retrieval quality were explained mainly by stemming aggressiveness, one might expect the best method simply to have the smallest output vocabulary. Table VIII contradicts that explanation. PyStemmer reduces the 6,046,958 no-stemming vocabulary to 5,432,989 outputs. The sealed Radixor model retains 5,614,856 outputs—181,867 more than PyStemmer, or approximately 3.35%—yet obtains the higher in-sample retrieval objective.

The workload is also nearly unchanged relative to PyStemmer. The sealed model evaluates about 41.762 million posting entries per query versus 41.781 million for PyStemmer (about 0.045% lower), and touches about 22.510 million candidate documents versus 22.514 million (about 0.017% lower). The measured effectiveness increase is therefore not explained by a materially larger candidate workload.

These observations address RQ6 for the methods compared. A scalar “stemming strength” statistic is insufficient. What matters is *which* occurrences are joined and separated. The sealed policy is simultaneously less aggressive in global vocabulary reduction and more effective for the measured relevance objective.

TABLE VII

DESCRIPTIVE ENGLISH PER-QUERY CONTRAST: SEALED RADIXOR MINUS PYSTEMMER. INTERVALS AND NOMINAL TWO-SIDED SIGN-FLIP VALUES ARE COPIED FROM THE SUPPLIED LANGUAGE-LEVEL PAIRED-EFFECT ROWS. THEY DO NOT ACCOUNT FOR POLICY SELECTION ON THESE QUERIES AND ARE NOT CONFIRMATORY INFERENCE. W/T/L DENOTES QUERY WINS/TIES/LOSSES.

Metric	Δ	Descriptive 95% interval	Nominal p	W/T/L
nDCG@10	+0.005174	[0.001888, 0.008853]	0.0024	49/741/9
Recall@100	+0.003692	$[-5.21 \times 10^{-7}, 0.008219]$	0.0904	8/789/2
Recall@1000	+0.001460	[0.000000, 0.004172]	0.5061	2/797/0
AP@1000	+0.003711	[0.000255, 0.007488]	0.0405	90/685/24
MRR@10	+0.005086	[0.000347, 0.010274]	0.0428	28/764/7

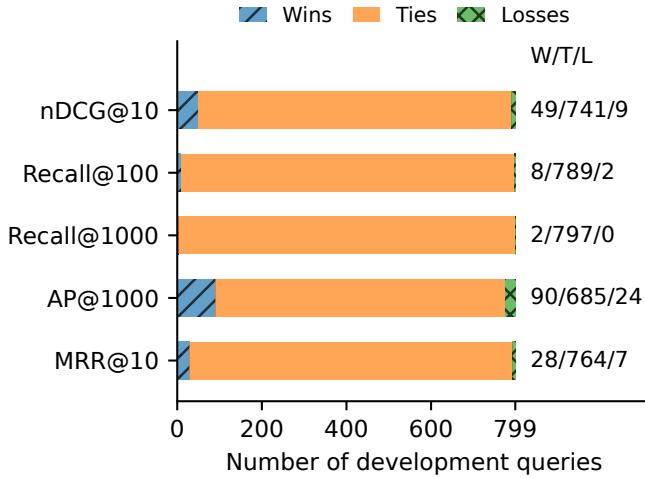


Fig. 5. Most evaluated queries tie. Counts include all 799 queries rather than enlarging only the changed subset. A metric tie need not mean identical retrieved documents. The full count scale guards against reading a small aggregate gain as a universal improvement. Source: English rows in data/pairwise-effects.csv.

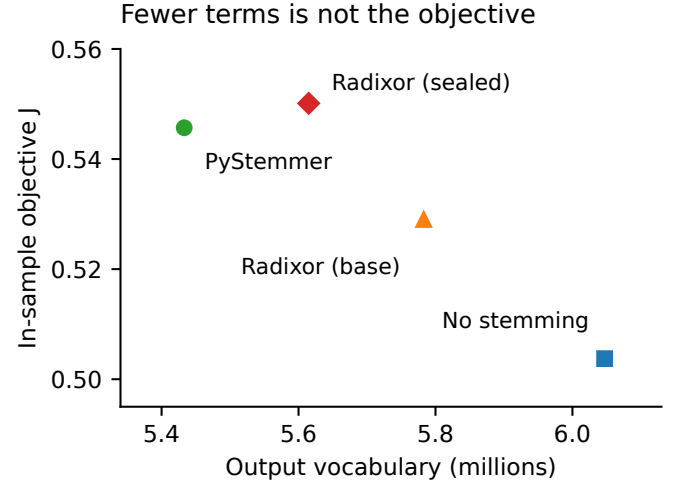


Fig. 6. Vocabulary reduction alone does not order the observed objectives. Sealed Radixor retains 181,867 more output terms than PyStemmer but attains a larger in-sample J . These four operating points are not a fitted response curve or a measured Pareto frontier. Axes are truncated for readability. Sources: final query and index summaries.

TABLE VIII

FULL VOCABULARY CENSUS AND MEAN DETERMINISTIC QUERY WORK. POSTINGS AND CANDIDATE DOCUMENTS ARE PER QUERY, IN MILLIONS (M). THESE ARE WORK COUNTS, NOT LATENCY MEASUREMENTS.

Method	Vocabulary	Postings	Candidates
NONE	6,046,958	41.160M	22.384M
RADIXOR-BASE	5,782,800	41.747M	22.496M
PYSTEMMER	5,432,989	41.781M	22.514M
RADIXOR-SEALED	5,614,856	41.762M	22.510M

E. Three Index Quantities That Must Not Be Collapsed

The saved census permits a more discriminating analysis than vocabulary reduction alone. Table IX separates the number of output terms, full-corpus term–document incidence, the fraction of token positions whose output string differs from their input, and bytes in the *query-selective* positional store. The four columns measure different objects.

Derived structural result. Relative to RADIXOR-BASE, the sealed model has 167,944 fewer output terms but 1,653,628 *more* term–document incidences. By Proposition 2, pure coarsening could not increase incidence. Pure refinement could not decrease the number of nonempty observed classes.

Therefore, conditional on the recorded common-corpus scalar semantics, the two observed partitions are *incomparable under refinement*: the sealed policy is neither merely a more aggressive merger nor merely a more conservative splitter. This deduction uses two preserved integer census columns and does not require inventing individual lexical examples.

Rewritten-position counts illustrate another pitfall. They are approximately 27.18% for PyStemmer, 14.75% for base Radixor, and 74.69% for sealed Radixor. It would be incorrect to interpret that last number as evidence of the strongest conflation. As Proposition 1 shows, a relabeling can rewrite every string while preserving all class memberships. The aggregate counters do not identify how much of the sealed rewrite count comes from relabeling, barriers, or class changes; a matched surface-to-output inventory is required for that decomposition.

Physical storage gives a third ordering. The sealed query-selective gzip store is approximately 1.871 GB, about 0.879% below PyStemmer’s 1.887 GB, despite its larger full term–document incidence and larger vocabulary. The stores cover different query-reachable class sets, and varint/gzip representation adds its own effects. Neither difference estimates the byte size of a fully materialized production index. A valid full-

TABLE IX

DISTINCT STRUCTURAL QUANTITIES FROM THE FINAL INDEX SUMMARIES. FULL INCIDENCE IS COMPUTED OVER THE COMPLETE CORPUS; THE GZIP STORE MATERIALIZES ONLY QUERY-REACHABLE POSTINGS AND USES DECIMAL GB. REWRITTEN-POSITION PERCENTAGE IS LABEL-SENSITIVE, NOT A COUNT OF CHANGED EQUIVALENCE RELATIONS. THESE ARE RECORDED STRUCTURAL VALUES, NOT STEMMER-MODEL SIZES OR LATENCY MEASUREMENTS.

Method	Output terms	Full $\sum_C df(C)$	Rewritten (%)	Query-store GB
NONE	6,046,958	1,526,613,164	0.00	1.697886
PySTEMMER	5,432,989	1,494,208,050	27.18	1.887252
RADIXOR-BASE	5,782,800	1,501,142,516	14.75	1.841400
RADIXOR-SEALED	5,614,856	1,502,796,144	74.69	1.870669

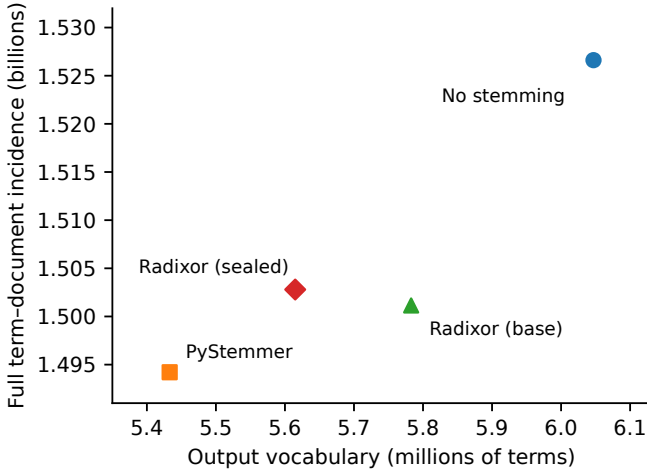


Fig. 7. Fewer classes need not mean fewer term–document incidences. From base to sealed Radixor, vocabulary decreases while incidence increases, ruling out a pure coarsening or refinement on the observed corpus. These are aggregate partition diagnostics; they do not reveal which pairs changed. Both axes are truncated.

index compression comparison would keep codec, document ordering, positions, metadata policy, and materialization scope fixed and report all corresponding components.

The occurrence total stays 2,196,437,823 for every final scalar method. Stemming reduces some repeated class–document memberships and alters labels; it does not delete the original positions in this benchmark. This is the relevant bridge to prior discussions of inverted-index size, not a claim that a vocabulary reduction percentage transfers directly into an equally large reduction in index bytes.

F. Workload Similarity Does Not Imply Identical Costs

The query-work archive distinguishes repeated posting-list accesses from unique lists and entries. These are useful controls: a stemmer can merge two query terms, change DF, and alter both repeated and unique work even when the number of original query positions is constant. Query-selective index size, mean candidate count, and mean posting entries are therefore complementary diagnostics rather than interchangeable efficiency metrics.

The near-equality of sealed and PyStemmer mean work supports a narrow finding: the measured gain is not accompanied by a large increase in these deterministic workload counts. It does not establish a per-query cost guarantee. Equal means

can conceal tail differences, and recorded medians or 95th percentiles do not identify paired worst-case regressions. Build-stage wall-clock fields are preserved as execution metadata, but single recorded runs with no controlled repeated timing design cannot support a general latency or throughput claim.

XI. RETRIEVAL DEPTH, OBJECTIVE WEIGHTS, AND WHAT THE SUMMARIES CAN RESOLVE

A useful evaluation separates three questions: how well the first results are ordered, how much judged-relevant material enters the candidate set, and how the policy was selected. The first two are different observations; the third governs what may be inferred from them. This section extracts the depth information available in the supplied summaries without inventing document-level trajectories.

A. Metric Definitions and Their Operational Meaning

Let R_q be the number of judged-relevant documents for query q , let $y_{q,r} \in \{0,1\}$ denote relevance at rank r , and assume $R_q > 0$ for an evaluated query. The standard recall quantity is

$$R_q @ K = \frac{1}{R_q} \sum_{r=1}^K y_{q,r}. \quad (22)$$

Its denominator is the judged-relevant count, not the cutoff K . Recall@100 is therefore not precision among 100 returned documents and not a measure of all unknown relevant documents in the corpus.

For binary judgments, nDCG@10 normalizes $\sum_{r=1}^{10} y_{q,r} / \log_2(r+1)$ by the ideal value for that query. It rewards early placement but is not precision@10. MRR@10 uses the reciprocal rank of the first judged-relevant hit within ten, or zero if none occurs. AP@1000 aggregates precisions at judged-relevant ranks up to 1000, with the judged-relevant total as denominator under the standard truncated-AP convention. The benchmark names and manifest specify the reported measures; an executable archival release must also pin evaluator versions and empty-relevance conventions.

All reported means average over queries. A query with two judged-relevant passages has the same averaging weight as a query with twenty. Consequently, differences below describe macro-averaged fractions, not counts of distinct passages and not micro-averaged recall. The original per-query runs, which are not supplied, are required to switch to those alternative estimands.

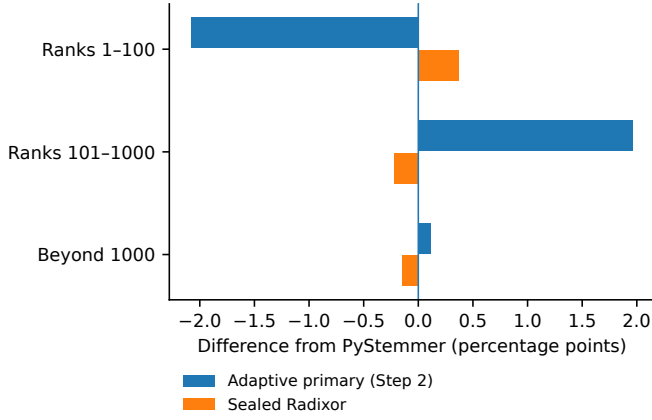


Fig. 8. Depth-band differences from PyStemmer at the common ranker. Adaptive primary has less judged relevance in the first 100 and more at ranks 101–1000. Sealed Radixor reverses that aggregate deficit. Each three-bar contrast sums to zero by arithmetic construction (with a coverage interpretation conditional on nonempty relevant sets); the plot does not identify passage transitions between bands.

B. An Exact Depth-Band Decomposition from Existing Recall Means

Two recall cutoffs are sufficient to compute three depth-band quantities. Interpreting them as fractions of judged-relevant sets requires every included query to have a nonempty judged-relevant set. The supplied aggregates do not expose a qrel-level verification of that condition. Without it, the arithmetic still holds for the reported means but is not a validated coverage partition. With bars denoting query means,

$$p_{1:100} = \overline{R@100}, \quad (23)$$

$$p_{101:1000} = \overline{R@1000} - \overline{R@100}, \quad (24)$$

$$p_{>1000} = 1 - \overline{R@1000}. \quad (25)$$

The last category includes judged-relevant documents absent from the stored run; it does not assign them a known exact rank. Linearity of the mean makes this decomposition exact under a shared query set and relevance denominator. These three shares sum to one by construction. That accounting identity is not a conservation law for nDCG, AP, utility, or “quality.”

The adaptive primary row leaves approximately 2.077 percentage points less judged relevance in ranks 1–100 than PyStemmer, with 1.962 points more in ranks 101–1000 and 0.115 points more beyond the run depth. Much of its observed Recall@100 deficit is therefore associated with relevant material present deeper in the ranking rather than a comparable loss at depth 1000. This is compatible with a depth-sensitive policy difference, but the aggregate data do not show which passages moved, whether the same passages were retrieved, or whether any particular change was morphological.

For sealed Radixor, the corresponding differences are +0.369 points in the first 100, −0.223 at ranks 101–1000, and −0.146 beyond 1000. These quantities make the title question measurable without drawing a fictitious smooth curve through nDCG, recall, and AP values. An analogous top-10 recall decomposition cannot be recovered from nDCG@10: different

rank patterns can have the same nDCG, and nDCG is not Recall@10.

C. What a True Rank-Transition Analysis Would Require

For each judged-relevant passage, a paired run could record its bands under two methods and form a transition matrix T_{ab} from old band a to new band b . The differences in Eq. (25) are merely its row/column marginal differences after query normalization. Many transition matrices share those marginals. One method could retrieve a different set of passages at the same cutoff, while another could move the same passages upward; the means alone cannot distinguish them.

A useful future diagnostic would separately count passages that enter the first stage, passages that cross a cutoff within an unchanged candidate set, and passages whose rank changes without crossing the chosen cutoff. The first is a coverage change; the second is an operational depth change; the third can still affect nDCG or AP. Recording these transitions is preferable to equating every Recall@100 improvement with broader linguistic coverage.

D. Objective Sensitivity Without Retraining

The Step 2 compromise can be quantified as a choice of weights. At the common point, adaptive primary minus PyStemmer has

$$(\Delta n, \Delta r, \Delta a) = (0.00300734, -0.02077064, -0.00333787), \quad (26)$$

for nDCG@10, Recall@100, and AP@1000. Consider the one-parameter family

$$J_\alpha = \alpha n + (0.9 - \alpha)r + 0.1a, \quad 0 \leq \alpha \leq 0.9. \quad (27)$$

For these *fixed policies and fixed runs*,

$$\Delta J_\alpha = \alpha(\Delta n - \Delta r) + 0.9\Delta r + 0.1\Delta a. \quad (28)$$

The sign changes at $\alpha \simeq 0.8002$: roughly 80% weight on nDCG, 10% on Recall@100, and 10% on AP. At the study’s $\alpha = 0.5$, the recall deficit dominates and PyStemmer has the larger composite objective (Fig. 9). This is an interpretable decision boundary, not a new optimization run.

The final sealed comparison is different: all five reported mean differences from PyStemmer are strictly positive. Consequently, *for these fixed final runs*, every nonnegative weighted sum of these five metrics with at least one positive weight favors the sealed model. This follows directly by multiplying positive differences by nonnegative weights and summing. It is a useful robustness statement about the observed means. It does not establish generalization, simultaneous confidence, superiority under unmeasured metrics, or performance of policies retrained for other weights.

The observed J gain over PyStemmer decomposes into approximately 0.002587 from nDCG@10, 0.001477 from Recall@100, and 0.000371 from AP@1000. These are contributions to a declared arithmetic objective, not a causal attribution to particular lexical edits. Assigning marginal benefit to each barrier would require additional ablations, including interactions among barriers.

TABLE X

MACRO-AVERAGED SHARES OF JUDGED RELEVANCE BY DEPTH, IN PERCENT, AT THE COMMON BM25 POINT 1.125/0.30. THE ADAPTIVE ROW IS FROM STEP 2 AND THE SEALED ROW FROM THE LATER FINAL EXPERIMENT; THEIR MODEL IDENTITIES REMAIN DISTINCT. ALL FIGURES ARE DERIVED FROM EXISTING RECALL MEANS; THE COVERAGE INTERPRETATION ASSUMES NONEMPTY RELEVANT SETS FOR EVERY INCLUDED QUERY.

Method	Ranks 1–100	101–1000	Beyond 1000
PyStemmer	82.915	9.126	7.958
Adaptive primary (Step 2)	80.838	11.088	8.074
Sealed Radixor	83.285	8.903	7.812

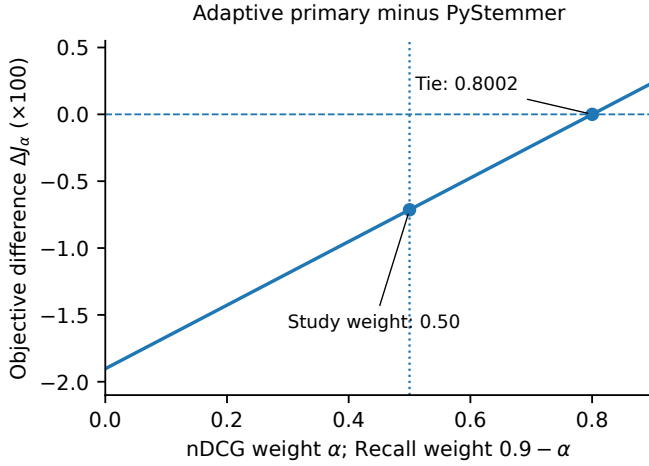


Fig. 9. Reweighting the same Step 2 common-point runs changes the preferred fixed policy. The line is an exact linear calculation from three reported means; it is not an optimizer response curve. Neither the dictionary nor BM25 is retained as α changes.

E. From an Observed Compromise to an Optimization Hypothesis

A rigid rule-based stemmer provides a complete policy as a starting point. An adaptive search constructs a policy through accepted moves. The difference is algorithmic: a useful group of edits may have no individually improving prefix. For example, moving a relevant document from rank 180 to 125 and then to 105 yields no Recall@100 reward until a further move crosses 100. These ranks are illustrative, not observations from MIRACL.

Four explanations for a shortfall should therefore be kept separate: objective mismatch, scorer mismatch, search-path restrictions, and representational or candidate-space restrictions. Full PyStemmer projection and subsequent positive materialization answer part of the representational question on the finite, query-reachable vocabulary. They do not prove universal equivalence on unseen forms. Swap operations address some non-improving single-edit paths but do not remove all search barriers.

Finding one policy with a top-rank/recall compromise does not establish a Pareto frontier. Such a claim requires a defined feasible set, separate or multi-objective searches, non-dominance checks under a declared budget, and validation independent of selection. The sealed example shows why that caution matters: the earlier compromise is not unavoidable among the policies eventually constructed for this development

set.

XII. HISTORICAL FINDINGS AND THE MISSING COMPARISON

A. A Methodological Explanation, Not a Universal Failure Story

Apparently conflicting findings can concern different interventions and different targets of inference. Changing a language-wide rule set, refining corpus-specific classes, and choosing per query whether to stem are not the same treatment. Likewise, intrinsic pairwise accuracy, first-page ranking, and deep candidate recall are not the same outcome. Equation (1) makes this distinction explicit. It explains how disagreement can arise without either measurement being wrong; it does not retrospectively identify the cause of every historical result.

The strongest inference from the present data is more specific: *the measured position of a default stemmer can be changed by adapting its lexical policy without replacing the engine.* Consequently, a negative or inferior default result does not show that the engine lacks a better realizable policy. This is an identification problem when a study generalizes from selected configurations to the attainable capabilities of a method. It is not a defect in studies that correctly restrict their conclusions to the configurations they tested.

An earlier concrete precedent, Xu–Croft’s corpus-conditioned grouping, shows why a universal claim that learned conflation was unavailable would be inaccurate [1]. Hull’s 1996 analysis also explicitly addressed statistical and per-query variation [4]. These observations do not establish an earlier equivalent of the Egothor patch engine. Nor can later HPS, CST or XSTEM publications displace the chronology of Galambos’s 2001 contribution merely because they describe some form of adaptation [2], [16]–[18]. Priority, overlap in a particular capability, and equivalence of complete systems are separate questions.

B. Why a Capability-Based Prior-Art Review Matters

Calling two systems configurable does not establish equivalent native transformations, learning procedures, or deployment behavior. The companion paper compares actual method descriptions, not abstract labels: it distinguishes HPS’s restricted output family, CST’s captured-span transformations, and XSTEM’s terminal replacements from the older Egothor patch language [6]. Its formal witnesses separate native actions without pretending that a more elaborate learner or an identical trie data structure implies the same deployment capability.

A concrete bibliographic gap is identifiable: XSTEM’s cited comparison omits the Egothor/Stempel lineage even though its trie-selected transformations were documented in Lucene before XSTEM’s 2022 submission [16], [18], [19]. That limits the coverage of that particular related-work account. A second, different gap is documented in the companion: CST cites the Polish Stempel distribution but does not identify Galambos/Egothor as the underlying algorithmic source [6], [17], [20]. A software citation is not a substitute for methodological attribution, and the language of its supplied dictionary is not the scope of its engine. These are concrete limitations in the dissemination and comparison of the earlier design. They do not, without reconstructed experiments, identify the cause of the broader literature’s retrieval results or replace an equal-budget comparison.

C. How Strongly the Results Can Be Stated

The final selected Radixor policy improves every recorded metric mean over both standard Radixor and PyStemmer. Thus the improvement over the standard model is not merely a choice of weights that rewards one measured metric while sacrificing another. Standard-model adaptation changes the deployment’s position in the measured multi-metric space. The earlier Step 2 results, by contrast, show mixed metric advantages and objective-dependent ordering. These are distinct models and stages, not inconsistent summaries of one run.

This difference supports a representation-level interpretation: the useful intervention is not simply to remove more endings or to shrink the vocabulary further. The index-incidence argument shows why the final transformation cannot be described as a pure coarsening or pure refinement of the standard partition. A flexible policy can relocate boundaries in both directions. Negative exclusions are therefore not a concession that invalidates positive adaptation; they address another side of the same matching relation.

The development-set selection remains consequential. The evidence is a constructive demonstration of attainable selected-task improvement and of a deployment obstacle that must be checked. It is not an unbiased prediction of transfer, an ablation of each intervention, or a measurement of superiority over other systems after comparable adaptation. These distinctions allow a strong conclusion about what was realized without turning it into a stronger claim about what has not been tested.

D. What Would Establish the Historical Hypothesis

To test whether a restricted policy menu explains a particular earlier negative result, first reconstruct its original collection, judgments, tokenization, scorer and stopping conditions. Then compare the original menu with a richer policy search under matched information and optimization budgets, selecting solely on training/development folds. A crossed scorer experiment distinguishes a representation limitation from miscalibration. A cross-implementation deployment of the same chosen partition distinguishes search quality from compiler fidelity. Only a held-out improvement attributable to these controlled changes would support that study-specific causal explanation. Accumulating such replications across a defined literature sample would be needed for a field-level explanation.

XIII. WHAT INCOMPLETE JUDGMENTS DO—AND DO NOT—PERMIT

A. The Annotation Pool Is Not the Retrieval Universe

MIRACL’s published annotation procedure retrieves candidate passages with BM25, mDPR, and mColBERT, combines normalized scores, and presents the top ten ensemble results for binary relevance assessment. The component runs retrieve to depth 1,000, and the construction also retains specified positive judgments from the preceding Mr. TyDi resource. Its published English development statistics list 799 queries and 8,350 judgments [14]. These are query–passage judgments, not a fully labelled corpus and not a count of distinct passage identifiers. The supplied run manifest remains the authority for the corpus indexed here; its exact qrels snapshot should additionally be pinned before full replay.

A new lexical policy may retrieve a relevant passage that the pooling systems did not expose to an assessor. Scoring it as unjudged/nonrelevant can conceal a genuine gain. Conversely, optimization can preferentially retrieve the small subset already marked relevant. The resulting benchmark score is observable and useful, but it is conditional on the judgment set. Work on incomplete IR evaluation motivates reporting this sensitivity rather than treating absent labels as complete semantic truth [21].

B. Why Not Keep Only Documents the Assessors Could Have Seen?

Restricting the collection to assessed or pooled documents appears to remove unknown relevance. It also changes the question. Let $\mathcal{D}$ be the original collection, P_q a query’s pool, and $A_q \subseteq P_q$ its assessed candidates. Three different experiments result:

Full retrieval: rank all of $\mathcal{D}$ with statistics computed on $\mathcal{D}$ and evaluate against the available judgments. This retains the first-stage search problem, with its incomplete-label limitation.

Pool-conditional ranking: keep the full-collection statistics but rank only P_q or A_q . This tests ordering within a pre-selected candidate set. It cannot measure whether a method discovers relevant documents outside that set, and its candidate distribution inherits the pool builders’ selection. It may be a useful supplementary reranking diagnostic, but not a substitute first-stage benchmark.

Reduced-collection retrieval: build a new index from a union of selected documents and recompute N , DF , and average length. This changes both the competitors and the scorer’s evidence. A gain or loss no longer isolates the stemmer’s effect on the original task. Even a document assessed for one query remains potentially unjudged for another.

The exact sets that an assessor *could* have seen cannot be inferred from the final qrels alone; reconstructing them requires the original candidate runs and pooling rule. Nor is the union of those candidates a completeness certificate for relevance. The correct answer is therefore not that a restricted experiment is forbidden, but that its estimand must be declared and kept separate. This paper reports the full-collection task. None of these restricted-candidate sensitivity experiments has been run here.

The query-selective posting store described earlier is different from all three restrictions on document eligibility. It retains every matching occurrence in the full collection for each needed term and keeps the original corpus statistics. It omits irrelevant *physical lists*, not eligible documents from the lists that are searched.

C. A Practical Sensitivity Package

The most direct follow-up is a blinded supplementary pool containing previously unjudged top-ranked passages from each frozen competing system. Sampling should not disclose which system contributed a passage. A second report can quantify judged-document coverage by cutoff and compare conclusions under an incompleteness-oriented measure such as bpref [21]. Neither step automatically establishes exhaustive relevance: added judging reduces the unknown region, and a different metric changes what is emphasized.

For the existing artifacts, only the limitation can be documented. Per-query runs are needed to identify unjudged exposures, and the judgment snapshot is needed to distinguish newly exposed relevant, nonrelevant, and unknown passages. Aggregated Recall@100 cannot supply those identities. Likewise, deleting unjudged results after observing which method benefits would introduce another adaptive choice rather than repair the original design.

D. Selection Bias Is a Separate Issue

Compiler leakage is a mismatch between an intended class and its executable realization. Judgment incompleteness is missing relevance information. Evaluation leakage is the use of evaluation information during model selection. They have different remedies: a closure audit, additional or sensitivity-aware assessment, and an independent evaluation design, respectively. Eliminating compiler leakage does not eliminate the other two.

The same development judgments guide BM25 selection, oracle moves, and final reporting here. Repeated selection on a finite evaluation criterion can overfit that criterion; applying a paired test after selection does not recreate an untouched experiment [15]. The provided confidence intervals and randomization summaries remain informative descriptions of the selected runs, with their selection caveat. They are not presented as confirmatory evidence that a future query population will benefit.

XIV. THREATS TO VALIDITY AND LIMITATIONS

A. In-Sample Oracle Optimization

The central limitation is deliberate and substantial: the MIRACL development qrels used to optimize the partition are the same judgments used to report the final effectiveness. The sealed advantage over PyStemmer is therefore not an unbiased estimate of future performance. The reported randomization *p*-values do not repair this selection effect [15]. The experiment establishes constructive feasibility and mechanism. A generalization claim requires a train/dev/test or cross-collection protocol.

B. Single Language, Collection, and Ranker

Only English MIRACL and BM25 are studied. The result does not establish that the same policy structure, barrier count, or effect size holds for other languages, domains, or rankers. MIRACL itself covers 18 languages [14], making multilingual follow-up possible, but those experiments are outside the present scope.

C. Incomplete Relevance Judgments

MIRACL provides extensive but finite judgments. As in standard pooled IR evaluation, unjudged documents are treated as nonrelevant by the reported metrics. A changed stemming policy can retrieve an unjudged document that is actually relevant. The qrels therefore define the measured objective, not complete semantic truth. Section XIII distinguishes full retrieval from pool-conditional and reduced-collection alternatives.

D. Objective Choice

Equation (17) emphasizes nDCG@10 and Recall@100 with a smaller AP@1000 contribution. Another application can rationally choose another objective, producing another partition. That dependence is a feature of the retrieval-policy view, but it limits universality of the specific oracle found here.

E. Observed-Vocabulary Seal

The fixed-point barrier scan covers its 6,046,942-surface inventory. The final no-stemming index census reports 6,046,958 terms, a difference of 16 that the supplied artifacts do not reconcile. We preserve both counts and restrict the zero-leakage claim to the audited inventory rather than assuming identical coverage. It does not prove that an arbitrary future unseen token cannot generalize into a protected class. A deployment that continuously ingests new vocabulary can periodically rerun the closure audit or use a compiler-native exact-boundary primitive.

F. Barrier Representation

The private-use marker and two-value guard are compact implementation devices chosen for the current Radixor compiler. They demonstrate realizability without altering the core algorithm, but they are not claimed to be the uniquely best representation. A native pinned-rule construct could encode the same policy more directly and may reduce model complexity further.

G. Artifact and Version Coverage

The final summaries and benchmark manifest are present. The actual sealed model, optimizer and sealer source revisions, full intermediate audit logs, terminal oracle checkpoint, and per-query ranked runs are not. The manifest identifies the sealed model by checksum but does not pin the exact PyStemmer/Snowball or default Radixor model version. These gaps limit independent replay; a checksum without its artifact is an identity reference, not a reproducible model distribution. The exact guard implementation and complete posting identity should be checked again against frozen source and model bytes before archival publication.

H. No Generic Wall-Clock Claim

The experiment reports deterministic posting-list and candidate-document work rather than end-to-end wall-clock latency across search engines. The workload figures show that the sealed effectiveness result is not obtained by a materially larger query-time candidate set, but they do not constitute a universal latency benchmark.

XV. DISCRIMINATING EXPERIMENTS AND JOINT OPTIMIZATION

The existing evidence motivates a concrete next-stage program rather than an unspecified call for more benchmarking. The following protocol is *prospective*. It defines what would distinguish the competing explanations developed above; it does not claim that these runs already exist.

A. Separate Objective, Scorer, Path, and Representation

Start each controlled arm from the same frozen corpus, tokenizer, dictionary seed, candidate relation, and optimization budget. Train separate policies for nDCG@10, Recall@100, and Eq. (17). Evaluate all resulting policies on the same complete metric vector, not only the metric that selected each policy. Repeat at a declared common BM25 point and under a matched per-policy tuning protocol. This identifies whether the earlier Recall@100 deficit responds to the objective, the scorer, or neither.

Next compare single-edit improvement search with bounded swaps or a beam of composite changes while preserving a common budget of exact state evaluations. A method that needs more proposed moves may otherwise receive a hidden advantage. Record accepted and rejected deltas, objective plateaus, class-membership changes, and full-state checkpoints. The cutoff example in Section XI predicts that composite moves can sometimes cross an uninformative plateau; it does not guarantee that a particular search strategy will do so on this corpus.

Finally, vary the seed policy: identity, the default Radixor model, the earlier adaptive model, and the full PyStemmer projection. Reusing PyStemmer as an initialization supplies a strong, already coherent policy. It is an empirical prior, not evidence that all of its relations are linguistically correct or irreducible. Every seed must face the same candidate-generation restrictions and an explicit accounting of additional preprocessing. If a desired finite partition can be specified explicitly but not materialized after validated sealing, the bottleneck is deployment semantics rather than the search objective.

A candidate vocabulary is another controlled factor. Include all known query-reachable class members when testing representation equivalence; then evaluate bounded candidate subsets separately. High collection-frequency coverage is not a substitute for including a rare, decisive query term. The reported 99.9998% CF coverage and subsequent value of full projection are precisely why this control matters.

B. Ranker–Policy Alternation and Its Stopping Claim

Let Π be a scalar lexical partition, $\theta = (k_1, b)$ the scorer parameters, and $J(\Pi, \theta)$ the chosen development objective. A blockwise procedure can be written as

$$\Pi_{t+1} = \mathcal{A}(\Pi_t; \theta_t, \mathcal{B}_\Pi), \quad (29)$$

$$\theta_{t+1} = \mathcal{T}(\theta_t; \Pi_{t+1}, \mathcal{B}_\theta), \quad (30)$$

where $\mathcal{A}$ and $\mathcal{T}$ are budgeted searches, not assumed exact argmax operators. Each should retain its incumbent unless a fully checked improvement exceeds a declared tolerance. After the final ranker step, the partition block must be tested again: a previously rejected conflation may have become useful under the new scorer.

A useful response record contains the complete tested parameter set, not only the winning point. Report common-support comparisons, near-best regions, and regret at a fixed operational point. The Step 2 result—different selected k_1 values but very small common-point regret—shows why convergence cannot be assessed merely by printing parameter changes. Plateaus and ties can make the selected parameter coordinate jump without a meaningful effectiveness change.

For a fixed finite observed vocabulary, a finite admissible partition family, and a finite scorer grid, strict improvement prevents revisiting a state and hence guarantees termination of the abstract accepted-state sequence. It does not guarantee a global optimum, and a budget limit can stop the search before even a local certificate is obtained. With continuous parameters, strict bounded improvement alone does not guarantee finite termination. An unchanged output of a heuristic merely shows stability under that heuristic’s tested moves. A claim of blockwise optimality additionally needs exhaustive or otherwise justified maximization of each declared block.

The experiment needed to test a *dictionary-induced optimum shift* is a matched response study of both the pre-update and post-update dictionaries on the same parameter support, followed by replication or validation of the apparent displacement. Comparing a learned model’s best point with its training anchor is insufficient. The existing Step 2 archive supplies a response profile for one learned dictionary and strong motivation for this experiment; it does not contain the complete alternating sequence.

C. Complexity and Stability Regularization

A deployment may prefer a smaller policy with nearly the same quality. One possible objective is

$$J_\lambda(\Pi, \theta, M) = J(\Pi, \theta) - \lambda C(\Pi, M), \quad (31)$$

with a fully specified complexity cost. Candidate costs include positive-override count, barrier count, distinct patch commands, executable model bytes, or a weighted number of changed lexical pair relations relative to a seed partition. These costs are not interchangeable. String-rewrite counts are label-dependent, while the partition relation is invariant to injective relabeling; a regularizer must choose the property it intends to control.

Here M is the compiled realization. A model-byte penalty belongs to that realization, not only the abstract partition, and a query-work penalty must state whether it concerns mean

work or a tail quantile. Parameter λ and the cost definition must be selected without using the final test labels. A family of observed non-dominated policies would support a measured effectiveness/complexity frontier under the declared search budget; the existing four final models do not characterize the entire attainable frontier.

D. Independent Evaluation After Development Reuse

The present 799 queries have already influenced multiple design and selection decisions. A retrospective split of them does not erase that exposure. For a confirmatory extension, use a genuinely untouched query set or new independent relevance assessment, freeze the analysis plan before examining its outcomes, and keep an auditable record of all prior label access. Nested selection can structure a newly designed study, but it cannot retroactively make previously inspected outcomes unseen [15].

Corpus-only vocabulary closure can be performed without test relevance labels, but using the deployment collection makes the procedure transductive with respect to that collection. Query-surface access, query-distribution access, and relevance-label access should be separately declared. Evaluating new queries on the same corpus tests query transfer; transferring the frozen policy to another corpus tests a different and stronger deployment condition. A second corpus-specific seal changes the deployment policy and must be reported as an additional adaptation, not as unchanged cross-collection transfer.

For multilingual replication, retain per-language effects and model identities instead of reporting only a pooled score. A lexical-family holdout can test generalization of learned transformations, whereas a query or topic holdout tests retrieval transfer. Neither substitutes for the other. The lexical-family holdout and pairwise-conflation studies in the earlier Radixor series [22], [23] and the extrinsic design here therefore require complementary, not interchangeable, controls.

E. Ablations That Would Test the Proposed Mechanism

A minimum mechanism package would preserve the full PyStemmer projection, explicit selected partition, unsealed compiled realization, and sealed realization as separate artifacts. Compare their positive mapping agreement, positional posting equality, score differences, and rank differences in that order. Then remove or add predefined barrier groups under a frozen checkpoint to estimate their contribution. Grouping by audited leakage mass, query exposure, or lexical pattern answers different questions and should be set before observing test outcomes.

Alternative barrier encodings can be compared under an identical target partition: marker/guard rules, exact exceptions, and a compiler-native boundary primitive. Measure closed-vocabulary correctness, open-vocabulary behavior, serialized size, heap usage, and stemming throughput separately. The current two-command construction establishes feasibility, not a minimum-size theorem. Likewise, the final query-work census supports an engineering comparison of this retrieval workload, not an end-to-end neural or RAG performance claim.

The experiment matrix in Appendix D records these arms and their current evidence status. Its purpose is to hold the scientific scope stable while distinguishing completed observations, mathematically derived consequences, and tests still needed to support predictive claims.

F. A True-Improvement Bound Is Not a Search-Cost Bound

If every accepted update increases the true objective by at least $\epsilon > 0$ and $0 \leq J \leq 1$, then $m\epsilon \leq 1 - J_0$, so at most $\lfloor (1 - J_0)/\epsilon \rfloor$ updates can be accepted. This bounds neither candidate evaluations nor their cost. An approximate acceptance test inherits the bound only if it guarantees the stated true improvement. The bound and the finite-vocabulary closure argument constrain different processes; neither certifies a global optimum.

XVI. CONCLUSION

The standard dictionary is not the performance ceiling of the Radixor engine. At the declared common scorer, the task-adapted model improves all five measured mean metrics and reverses the standard model’s ordering relative to PyStemmer. The gains over standard Radixor include 3.667% nDCG@10, 4.043% Recall@100, and 5.340% AP@1000. This establishes a realizable in-sample improvement, not generalization from labels that were reused for selection.

The representation change is not a scalar increase or decrease in stemming strength. Its lower output vocabulary and higher term–document incidence prove non-nesting under the stated scalar semantics. The earlier BM25 response study independently of that final model shows why parameter coordinates, near-optimal regions, metric weights, and query-level support must be distinguished.

The defensible historical conclusion is methodological: a default-policy contrast is not a family-level estimate, and different policy/ranker/population/objective combinations need not estimate one invariant treatment. It is not necessary to claim that prior researchers lacked adaptive methods in order to make this argument. Radixor’s contribution here is to make the policy intervention and its deployed result explicit, inspectable, and measurable. The next decisive evidence is matched-budget, out-of-sample comparison of adaptation procedures, not a stronger adjective attached to the present development outcome.

DATA, CODE, AND RELATIONSHIP TO THE COMPANION PAPER

The source package contains the original final summaries and manifest, the 164 Step 2 evaluations, labelled intermediate reports, generated tables and vector plots, and the analysis scripts. It rebuilds this article without network access after standard LaTeX dependencies are installed. The original optimizer, complete posting audit inventories, sealed-model bytes, and ranked runs are unavailable in the supplied material; their absence is not filled with synthetic records. The model hash identifies an expected artifact, not a distributed executable. The two papers share these data: Paper 8 owns the effectiveness analysis, while Paper 9 owns the realization contract and technical capability comparison. Repeated contextual numbers are not additional experiments.

APPENDIX A ORACLE SEARCH PSEUDOCODE

The implementation uses caches, bounded exact shortlists, and native sparse scoring for efficiency. The following pseudocode states the logical search independent of those engineering details. The state is a set of selected candidate edges whose connected components define the current conflation partition. Every accepted operation is evaluated under the frozen BM25 parameters and Eq. (17). The actual search starts from the complete PyStemmer projection rather than from identity.

Oracle partition search (logical specification)

- 1: Construct full query-reachable PyStemmer classes.
- 2: Initialize selected edges E realizing those classes.
- 3: Score $\Pi(E)$ under fixed BM25 and store the incumbent.
- 4: **repeat**
- 5: Shortlist ADD and REMOVE operations.
- 6: Include bounded REMOVE+ADD swaps when scheduled.
- 7: Evaluate shortlisted partition changes exactly.
- 8: Accept a tested change only if $\Delta J > \epsilon$.
- 9: Checkpoint and periodically rescore the full state.
- 10: **until** the search budget or declared stopping rule is reached
- 11: Return the best observed state, not a certified optimum.

The supplied package does not freeze the exact shortlist budgets, tolerance, or optimizer source revision. This pseudocode documents the inherited search logic, not an executable reproduction of its trajectory.

APPENDIX B MEASUREMENT INVARIANTS

The supplied manifest and index summaries record several invariants that keep lexical-policy comparisons interpretable. All methods process the same 32,893,221 documents in the same order and have the same SHA-256 digest of logical document identifiers and original document lengths. These records support shared N , original document lengths, and average document length; a document digest does not certify equality of query text. Shared normalized query text is a separate protocol requirement, supported here by the declared query set rather than an independently supplied query-text digest. The intended intervention changes the terms attached to original positions.

For the scalar methods in the final comparison, each original position emits exactly one term. Hence the total emitted term occurrence count remains 2,196,437,823 for NONE, PyStemmer, default Radixor, and sealed Radixor. A term’s TF in a document is the number of distinct original positions carrying it. This prevents a lexical rewrite from changing BM25 document length merely because it changes the output vocabulary.

Query-side processing follows the same analyzer as document-side processing. The final sealed model exposes 1,684 query-reachable output terms, compared with 1,687 for PyStemmer, 1,716 for default Radixor, and 1,837 without stemming. These counts are not effectiveness metrics; they

are useful checks that the query analyzer and index use the intended model identity.

APPENDIX C METRIC-SPECIFIC LOCATIONS IN THE STEP 2 SAMPLES

Table XI selects each metric’s maximum from the 41 archived rows for its method. The rows were generated by the original composite-objective search, so they do not constitute equally budgeted metric-specific optimization. Near ties and unvisited points prevent treating these locations as unique population optima. The original CSVs retain the individual metric values and search-stage identifiers for independent selection checks.

APPENDIX D A CONTROLLED EXPERIMENT MATRIX

Table XII turns the proposed explanations into distinguishable interventions. A result is “archived” only when it is actually supplied as a machine-readable summary; inherited intermediate observations remain separately identified. Availability is not a claim of independent replay.

APPENDIX E MINIMUM REPLAY AND AUDIT RECORD

Independent replay requires corpus/qrels revisions, normalized queries, tokenizer and analyzer versions, the sealed-model bytes, compiler settings, and retrieval/evaluation code. The optimizer must additionally supply candidate edges, exact budgets, seeds, tolerances, accepted checkpoints and the terminal partition. Query-level metrics and top-1,000 runs are needed for new paired inference and rank-transition analysis.

The compilation record must retain positive assignments, both guards, the audited inventory, leakage members, posting comparisons and closure iterations. The 128-entry positive-map discrepancy and 16-member vocabulary discrepancy require explicit member reconciliation. The companion systems paper specifies the complete audit record [6]. This package preserves the available summaries and regenerated analyses; it does not supply the missing executable replay inputs.

TABLE XI

BEST OBSERVED (k_1, b) BY METRIC WITHIN EACH METHOD'S STEP 2 SAMPLES. THE TABLE IS A SECONDARY CALCULATION FROM THE ARCHIVED ROWS, NOT FIVE NEW TUNING RUNS. THE COMPOSITE-OBJECTIVE LOCATIONS ARE IN TABLE II.

Method	nDCG@10	R@100	R@1000	AP@1000	MRR@10
NONE	1.200/0.30	1.125/0.30	1.275/0.35	1.050/0.30	1.200/0.30
PySTEMMER	1.050/0.30	1.125/0.25	1.500/0.40	1.050/0.30	1.050/0.30
ADAPTIVE-PRIMARY	0.975/0.30	1.050/0.30	1.200/0.40	1.125/0.30	1.125/0.30
ADAPTIVE-ALL	1.050/0.30	1.350/0.30	1.200/0.40	1.125/0.30	1.125/0.30

TABLE XII

DISCRIMINATING EXPERIMENTS AND CURRENT EVIDENCE STATUS. ALL PROPOSED ARMS REQUIRE FROZEN IDENTITIES AND A DECLARED EVALUATION BUDGET. A PROSPECTIVE HOLDOUT MUST BE GENUINELY UNEXPOSED, NOT A RETROSPECTIVE RELABELLING OF PREVIOUSLY USED DEVELOPMENT QUERIES.

Question	Controlled intervention	What would distinguish the explanation?	Present status
Scorer sensitivity	Same dictionary; vary only BM25	Response regions and common-point regret, not parameter displacement alone	164 Step 2 rows archived.
Objective mismatch	Same seed/candidates; train for nDCG, R@100, or J	Recall gain under direct recall training without changing scorer or budget	Fixed-run reweighting derived; training arms not supplied.
Search-path restriction	Single-edit versus composite-move search	Improvement under matched exact-state evaluation budgets	Swap-based search described; matched ablation absent.
Representation / candidates	Full finite target partition versus bounded candidate vocabulary	Exact target mapping possible after compilation and closure	Projection and materialization reported; source replay absent.
Compiler spillover	Explicit partition, positive-only compiler, then barriers	Positive agreement with posting mismatch before repair; equality after repair	Intermediate audit reported; final summaries archived.
Boundary importance	Predeclared groups removed from the same sealed checkpoint	Group and interaction effects on unseen queries	Not supplied.
Coupled optimization	Matched alternating policy/scorer updates	Stable or improving response regions after both blocks are retested	One adaptive response stage; full sequence absent.
Transfer	Freeze on development; evaluate new queries/corpus/languages	Improvements not selected on their evaluation labels	Not supplied.
Annotation sensitivity	Blinded added judgments and declared pool-conditional diagnostics	Robustness to unknown relevant passages and pool selection	Not supplied.
Deployment cost	Same target policy under alternate barrier implementations	Equal semantics with measured bytes, heap, throughput, and tail work	Aggregate final index/work counts only.

REFERENCES

- [1] J. Xu and W. B. Croft, “Corpus-based stemming using cooccurrence of word variants,” *ACM Transactions on Information Systems*, vol. 16, no. 1, pp. 61–81, 1998.
- [2] T. Brychcín and M. Konopík, “HPS: High precision stemmer,” *Information Processing & Management*, vol. 51, no. 1, pp. 68–91, 2015.
- [3] G. Göksel, A. Arslan, and B. T. Dinger, “A selective approach to stemming for minimizing the risk of failure in information retrieval systems,” *PeerJ Computer Science*, vol. 9, p. e1175, 2023.
- [4] D. A. Hull, “Stemming algorithms: A case study for detailed evaluation,” *Journal of the American Society for Information Science*, vol. 47, no. 1, pp. 70–84, 1996.
- [5] R. Nogueira and K. Cho, “Passage re-ranking with BERT,” 2019, revised 2020.
- [6] L. Galambos, “From lexical edits to verified stemming policies: Two-sided adaptation in radixor,” 2026, radixor Research Series, Paper 9. Companion manuscript, 20 September. Same experimental case; not an independent replication.
- [7] M. F. Porter, “An algorithm for suffix stripping,” *Program*, vol. 14, no. 3, pp. 130–137, 1980.
- [8] C. D. Paice, “Method for evaluation of stemming algorithms based on error counting,” *Journal of the American Society for Information Science*, vol. 47, no. 8, pp. 632–649, 1996.
- [9] F. N. Flores and V. P. Moreira, “Assessing the impact of stemming accuracy on information retrieval – a multilingual perspective,” *Information Processing & Management*, vol. 52, no. 5, pp. 840–854, 2016.
- [10] N. Ferro and G. Silvello, “Toward an anatomy of ir system component performances,” *Journal of the Association for Information Science and Technology*, vol. 69, no. 2, pp. 187–200, 2018.
- [11] D. Li and E. Kanoulas, “Bayesian optimization for optimizing retrieval systems,” in *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, 2018, pp. 360–368.
- [12] S. Wang, S. Zhuang, and G. Zuccon, “Large language models for stemming: Promises, pitfalls and failures,” in *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2024.
- [13] S. E. Robertson and H. Zaragoza, “The probabilistic relevance framework: Bm25 and beyond,” *Foundations and Trends in Information Retrieval*, vol. 3, no. 4, pp. 333–389, 2009.
- [14] X. Zhang, N. Thakur, O. Ogundepo, E. Kamaloo, D. Alfonso-Hermelo, X. Li, Q. Liu, M. Rezagholizadeh, and J. Lin, “Miracl: A multilingual retrieval dataset covering 18 diverse languages,” *Transactions of the Association for Computational Linguistics*, vol. 11, pp. 1114–1131, 2023.
- [15] G. C. Cawley and N. L. C. Talbot, “On over-fitting in model selection and subsequent selection bias in performance evaluation,” *Journal of Machine Learning Research*, vol. 11, pp. 2079–2107, 2010. [Online]. Available: <https://www.jmlr.org/papers/v11/cawley10a.html>
- [16] L. Galambos, “Lemmatizer for document information retrieval systems in JAVA,” in *SOFSEM 2001: Theory and Practice of Informatics*, ser. Lecture Notes in Computer Science, vol. 2234. Springer, 2001, pp. 243–252. [Online]. Available: https://doi.org/10.1007/3-540-45627-9_21
- [17] B. Jongejan and H. Dalianis, “Automatic training of lemmatization rules that handle morphological changes in pre-, in- and suffixes alike,” in *Proceedings of ACL-IJCNLP*. Association for Computational Linguistics, 2009, pp. 145–153. [Online]. Available: <https://aclanthology.org/P09-1017/>
- [18] K. Baker, “XSTEM: An exemplar-based stemming algorithm,” 2024, version 2, 2 June 2024; first submitted 2022. Preprint. [Online]. Available: <https://arxiv.org/abs/2205.04355v2>
- [19] Apache Lucene Project, “StempelStemmer.java, lucene 4.10.4,” 2015, source documents Egothor provenance and language-independent core; accessed 20 September 2026. [Online]. Available: <https://github.com/apache/lucene-solr/blob/69175ad4fe8d224ad6a348d05f686105999a92e8/lucene/analysis/stempel/src/java/org/apache/lucene/analysis/stempel/StempelStemmer.java>
- [20] A. Bialecki and Apache Lucene Project, “Stempel: Algorithmic stemmer for polish language,” 2015, overview shipped with Lucene 4.10.4; describes the earlier Egothor design and cites Galambos’s publications. [Online]. Available: <https://github.com/apache/lucene-solr/blob/69175ad4fe8d224ad6a348d05f686105999a92e8/lucene/analysis/stempel/src/java/overview.html>
- [21] C. Buckley and E. M. Voorhees, “Retrieval evaluation with incomplete information,” in *Proceedings of the 27th Annual International ACM SIGIR Conference*, 2004, pp. 25–32.
- [22] L. Galambos, “Does a Lexicon-Trained Stemmer Generalize? A 20-Language Study of Held-Out Lexical Families,” ResearchGate, 2026, Radixor Research Series, Paper 1. Author-released research manuscript. [Online]. Available: <https://doi.org/10.13140/RG.2.2.34383.85921>
- [23] —, “Classifying Multilingual Stemmers by Pareto Strata: Error Regimes in Pairwise Lexical Conflation,” ResearchGate, 2026, Radixor Research Series, Paper 3. Author-released research manuscript. [Online]. Available: <https://doi.org/10.13140/RG.2.2.25995.25126>