Cairo University at the CLEF 2026 JOKER Track

Heuristic Pun-Morphology Features and Tree Ensembles for Humor-Aware Information Retrieval

Fatimah Emad Eldin[1,*]

[1]Cairo Univercity, Giza, Egypt

Abstract

JOKER 2026 Task 1 asks systems to retrieve short humorous English documents that are both topically relevant to a query and instances of wordplay. This is challenging because topical lexical matches often include non-humorous encyclopedic or definitional distractors. We address the task with a two-stage retrieve-and-rerank pipeline that combines BM25 and dense bi-encoder candidate generation with a tree-based reranker using retrieval, lexical, phonetic, character-level, joke-format, and pun-morphology features. Our best submitted run achieves MAP=0.6140 on the official held-out leaderboard and ranks second among the submitted systems, showing that explicit joke-format and pun-morphology features are effective under very limited task supervision.

Keywords

information retrieval, wordplay, pun detection, learning to rank, feature engineering, tree ensembles, JOKER, CLEF 2026

1. Introduction

The JOKER track at CLEF [1] promotes the systematic evaluation of computational methods for processing wordplay across languages. Task 1, *Humor-aware Information Retrieval*, asks systems [2] to retrieve short humorous documents—typically puns or wordplay—given a short query, where retrieved texts must be both topically relevant to the query and instances of wordplay. The task is an English-only retrieval problem with a fixed corpus of ∼97k documents and a small set of training queries with manually annotated relevance judgments.

We participated in the English variant of Task 1, with the goal of maximising Mean Average Precision (MAP), the official primary metric. This paper reports a system that ranks second on the Codabench leaderboard with MAP=0.6140, a substantial gain over the publicly disclosed BM25+Anserini baseline of 0.1087 and over the previously best-published participant submission of 0.5915.

Main strategy. Our system is a two-stage retrieve-and-rerank pipeline. The first stage runs eight diverse retrievers in parallel (three BM25 variants and five dense bi-encoder models), takes the top-300 documents per retriever per query, and merges them into a candidate pool of approximately 290 documents per query. The second stage extracts a 75-dimensional feature vector for each (query, document) pair and trains a tree-based learning-to-rank ensemble on the small training set of 10 queries with 25 positive labels.

Key findings. Through ablations, we establish three findings.

1. **Joke-format features dominate.** When 25 positive labels are all the supervision available, dense semantic features (which require learning) contribute little, while shallow heuristic features detecting joke morphology—template openers (*Why did/What do you call*), punctuation patterns, hyphenated wordplay, and document length—contribute +0.33 MAP over a generic IR feature baseline.

CLEF 2026 Working Notes, 21 – 24 September 2026, Jena, Germany

*Corresponding author.

12422024441586@pg.cu.edu.eg (F. E. Eldin)

2. **Cross-year augmentation hurts.** The 2025 edition of JOKER Task 1 in English provides 660 additional positive labels. Naive augmentation *degrades* test MAP by 0.02–0.04 because the 2025 task has a different relevance distribution (conceptual queries) than the 2026 task (lexical queries containing the query word in 94% of relevant documents).
3. **Pool expansion gives a final boost.** Expanding the candidate pool from top-120 to top-300 raises the achievable ceiling and contributes another $+0.015$ MAP, but with diminishing returns. Beyond that, simple model ensembling does not improve on the strongest single model.

Reproducibility. All code, feature extraction pipelines, trained models, and Codabench submission archives are publicly available in the project repository.[1]

2. Background

2.1. Task definition

JOKER 2026 Task 1 defines an ad-hoc retrieval task [2]. Each query q is a short English string (usually a single word). Given the corpus $\mathcal{C}$, the system must output, per query, a ranked list of up to 1,000 documents from $\mathcal{C}$ by score. The relevance criterion is dual: a document is relevant only if (a) it is topically related to q and (b) it is itself an instance of wordplay. Evaluation follows TREC conventions: relevance labels are binary $\{0, 1\}$, and the primary metric is MAP over all test queries, supplemented by NDCG@{5,10,100}, Mean Reciprocal Rank (MRR), and Precision@10.

Input/output format.

An example training tuple, taken directly from the official release:

- `query` = `"abrupt"`
- `relevant document` = `"My friend's goodbye was so abrupt, he just said 'a-' and left!"`

Note that the query word `abrupt` appears literally in the document and the document is a pun (the punchline plays on the truncation of the word itself).

2.2. Dataset and exploratory data analysis

Corpus. The English corpus contains 96,603 documents. Document length is short (median 16 words, 94 characters; 90th percentile 32 words), but the corpus is heterogeneous: short jokes co-exist with encyclopedic dictionary-style entries, definitions, and Wikipedia-style prose passages. A purely length-based heuristic does not separate the two genres cleanly. Table 1 summarises the dataset statistics.

Queries. The released training set is unusually small: only 10 queries with 25 positive qrels in total (mean 2.5 positives per query). The training queries are alphabetically consecutive single words starting with the letter "a"[2], which makes the train sample structurally non-representative of the diverse 972 test queries (e.g. `vinous`, `rima oris`, `snippety`, `cyberspace`). There is no query-string overlap between train and test sets.

Lexical and humorous relevance. Humour-aware retrieval combines two relevance dimensions that are usually treated separately in standard IR: topicality with respect to a query and the presence of humour or wordplay in the retrieved text. In the JOKER setting, relevant texts are often short puns, riddles, one-liners, or joke-like utterances, while non-relevant corpus items may still share vocabulary

[1] GitHub repository.
[2] Specifically: `a`, `ability`, `abrupt`, `accord`, `accordance`, `acerb`, `acerbic`, `acid`, `acidic`, `acidulous`.

Table 1
Dataset statistics for the English variant of JOKER 2026 Task 1.

Statistic	Value
Corpus documents	96,603
Median document length (words)	16
Median document length (characters)	94
90th-percentile document length (words)	32
Documents containing "?"	8,160 (8.4%)
Documents ending with "?"	4,745 (4.9%)
Documents ending with "!"	3,188 (3.3%)
Training queries (with qrels)	10
Training positive labels	25
Mean positives per train query	2.5
Test queries (no released qrels)	972
Total relevant in test set (per Codabench)	4,572
Mean positives per test query	4.7
Train query length (median words)	1
Test query length (median words)	1 (max 4)
Train/test query string overlap	0

with the query but lack a humorous or wordplay interpretation. This makes the task different from ordinary ad-hoc retrieval over prose documents: exact lexical matching can identify many topically plausible candidates, but the final relevance judgment also depends on whether the candidate instantiates a punning or humorous form.

2.3. Related work

Wordplay and pun benchmarks. Computational work on puns treats wordplay as deliberate lexical ambiguity, often involving polysemy, homonymy, homophony, or near-homophony. SemEval-2017 Task 7 established a widely cited English benchmark for pun detection, pun location, and pun interpretation, separating the question of whether a sentence contains a pun from the finer-grained tasks of identifying the punning word and recovering its alternative meaning [3]. This line of work is relevant to humour-aware retrieval because a retrieved item must not merely mention the query topic; it must also contain the linguistic ambiguity or incongruity that makes the text humorous.

JOKER datasets. The JOKER shared-task series extends pun and wordplay processing to multilingual evaluation settings and includes tasks on detection, classification, translation, and retrieval [4, 5, 1]. JOKER 2024 introduced Task 1 as humour-aware information retrieval: systems were given a topic query and asked to retrieve short humorous texts, with relevance requiring both topical relatedness and wordplay. The 2024 English collection contained 61,268 documents, including 4,492 humorous texts, and the task received 26 submitted runs from 10 teams [6]. JOKER 2025 kept the same broad retrieval formulation while extending the setting to English and European Portuguese, with 77,658 English documents, 45,126 Portuguese texts, and language-specific relevance judgments [7]. Across these datasets, the central evaluation problem is not only finding documents about a query, but finding documents that are simultaneously query-relevant and instances of humour or wordplay.

Retrieval methodologies. Existing humour-aware retrieval systems commonly use a retrieve-then-filter or retrieve-then-rerank architecture. First-stage retrieval is typically handled by lexical models such as BM25 [8], dense sentence/document embeddings such as Sentence-BERT, E5, BGE, or GTE [9, 10, 11, 12], or hybrid combinations of lexical and neural retrieval. Later stages then try to distinguish genuinely humorous or punning candidates from merely topical candidates. In the JOKER

participant literature, the University of Amsterdam system for JOKER 2024 used standard ranking methods together with a pun-classification filter [13]; Chen et al. [14] combined retrieval with relevance-aware classification; Kuzmin [15] explored a hybrid BM25, multilingual-E5, and cross-encoder pipeline; and Vachharajani [16] used Qwen-family LLMs for humour filtering, retrieval, and translation.

Humour and pun classification. Work on humour classification provides useful signals for retrieval because topical similarity alone does not guarantee humorous relevance. Delabastita's typology of wordplay and translation strategies describes how puns can be preserved, replaced, omitted, or converted into non-punning renderings across languages [17]. More recent JOKER systems apply neural classifiers to humour genre and technique prediction: Narayanan et al. [18] fine-tune RoBERTa and BERT-uncased models for humour classification, while retrieval-oriented systems use pun or humour classifiers as downstream filters for candidate lists [13]. These approaches suggest that surface form, semantic ambiguity, phonological similarity, and learned humour labels can all contribute to identifying wordplay-bearing texts.

Learning-to-rank models. Modern IR commonly separates candidate generation from reranking, using richer features or models at the second stage than would be affordable over the full corpus [19]. Learning-to-rank methods such as RankNet, LambdaRank, and LambdaMART provide a standard framework for optimizing ranked output rather than independent document labels [20]. Tree-ensemble implementations and related gradient-boosting models, including LightGBM and XGBoost, are widely used for feature-based ranking and classification because they can combine heterogeneous lexical, neural, and metadata features [21, 22]. Random forests and scikit-learn gradient-boosting models provide additional pointwise baselines for small or feature-rich ranking datasets [23, 24].

Dense retrieval models. Neural dense retrieval represents queries and documents in a shared embedding space and ranks candidates by vector similarity. Sentence-BERT introduced siamese and triplet-network sentence encoders suitable for semantic similarity search [9]; later embedding families such as E5, BGE, and GTE use large-scale contrastive or multi-stage training to improve general-purpose text retrieval [10, 11, 12]. In humour-aware retrieval, such models can help retrieve conceptually related jokes even when lexical overlap is weak, but they may still require a separate humour-sensitive reranking or filtering component.

Phonetic and surface-form features. Puns often rely on sound similarity, spelling variation, affixation, punctuation, and formulaic joke structures. Phonetic encodings such as Soundex, Metaphone, or NYSIIS are therefore commonly useful feature families for approximating homophony or near-homophony, especially in English wordplay. Surface-form features, including question marks, punchline punctuation, hyphenation, quotation, short length, and recurring joke templates, provide complementary evidence about whether a candidate text has the structure of a joke rather than an encyclopedic or definitional passage.

3. System Overview

Our system is a two-stage retrieve-and-rerank pipeline, illustrated in Figure 1. Stage 1 produces a candidate pool of approximately 290 documents per query. Stage 2 reranks them using a tree-based learning-to-rank model trained on 75 features per (query, document) pair.

3.1. Stage 1: First-stage retrieval

We deploy eight first-stage retrievers covering both lexical and dense semantic signals.

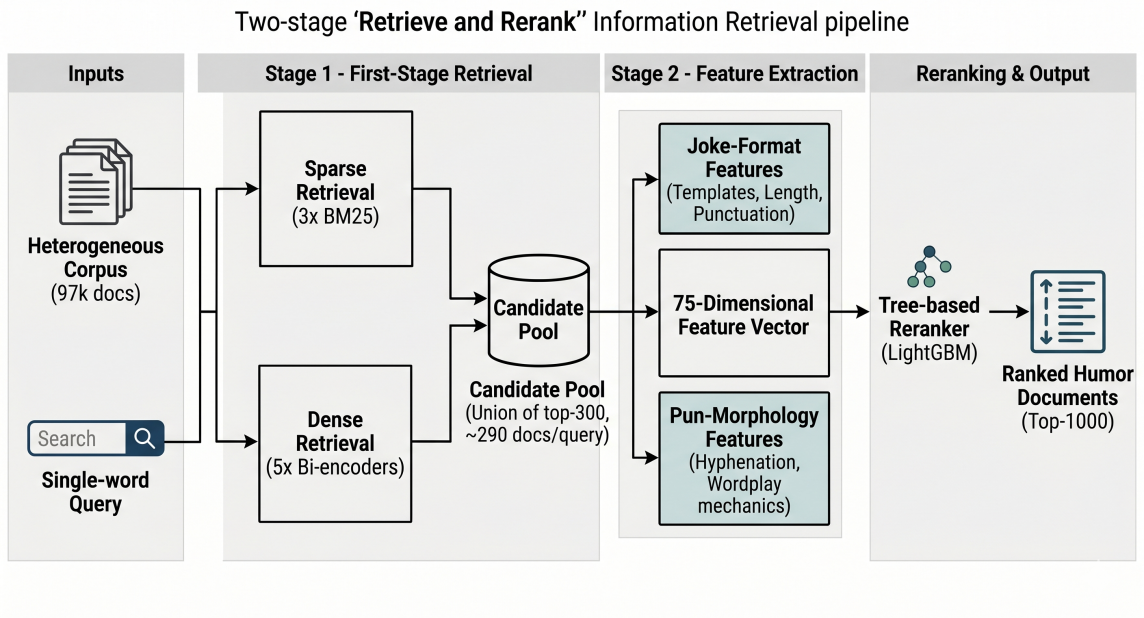

Figure 1: Two-stage pipeline. Stage 1 maximises recall through retriever diversity; Stage 2 reranks with hand-crafted features that distinguish jokes from non-jokes.

BM25 variants. We index the corpus with three BM25 variants over different token streams: (a) lemma-tokenised with $k_1 = 1.5, b = 0.75$; (b) lemma-tokenised with $k_1 = 1.2, b = 0.50$; (c) Porter-stemmed with $k_1 = 1.5, b = 0.75$. Stopwords are removed.

Dense retrievers. We use five bi-encoder models without any task-specific fine-tuning: E5-base-v2 and E5-large-v2 [10], BGE-base-en-v1.5 and BGE-large-en-v1.5 [11], and GTE-base [12]. Documents and queries are L2-normalised; similarity is cosine.

Candidate pooling. For each query, we take the union of the top-300 documents returned by each retriever (with documents from training-set relevance labels added for training queries). The resulting per-query pool averages 290 documents.

3.2. Stage 2: Feature extraction

Each (query, document) pair is described by 75 features organised into seven groups, summarised in Table 2.

3.2.1. Joke-format features

These features fire on the surface signatures of jokes regardless of query identity. They detect:

- Template openers: `Why did/do/does`, `What do you call`, `What's the difference between`, `Knock knock`, `I used to`, `A <noun> walks into. . .`
- Punctuation signals: presence/count/position of `?` and `!`, two-sentence document structure (classic Q-then-A puns)
- Length signals: indicator for ≤ 30 words, indicator for ≤ 15 words
- Capitalisation play: counts of intra-word capitalisation and all-caps tokens

A composite `jfmt_score` aggregates the strongest signals:

$$\textsc{jfmt\_score}(d) = 0.35\,T + 0.20\,O + 0.15\,E + 0.15\,Q + 0.10\,S + 0.05\,\min(1, N) \tag{1}$$

Table 2
Feature groups in the reranker (75 features total). The two right-most groups (joke format and pun morphology) are the empirical contribution of this work for puns and contribute the bulk of the gain over generic IR features.

Group	Count	Examples
Retrieval scores	24	raw score, rank, and reciprocal-rank for each of the 8 first-stage retrievers
Dense cosines	5	explicit cosine $\langle q, d \rangle$ for each dense model (re-computed from cached embeddings rather than relying on stage-1 ranks alone)
Lexical overlap	6	query-term recall over lemma and stem sets, Jaccard, query-substring match, query token count in doc
Phonetic	4	Soundex, Metaphone, NYSIIS match between query and any doc token
Character / morphology	4	character-3-gram Jaccard and recall, minimum Levenshtein distance to any doc token
Statistical	7	doc length (words, chars), query length, query/doc length ratio, query average/max IDF, first-occurrence position of query word (raw and normalised)
Joke format (jfmt)	15	template match (e.g. `"Why did/What do you call/Knock knock/I used to"`), punctuation patterns (`?`, `!`, ends-with), question-opener, sentence count, intra-word capitalisation, all-caps words
Pun morphology (adv)	10	hyphenated word count, single-letter-dash count (the `"a-rupt"` pattern), intra-word caps, repeated characters (`"AHHHHH"`), quoted speech, composite wordplay score, query-as-prefix/suffix in hyphenated words

where T=template match, O=starts with a question word, E=ends with "!", Q=has "?" followed by punchline text, S=document is short ($\leq$ 30 words), and N=count of "?" characters. We use this composite for interpretability; the tree models also receive the constituent indicators individually.

3.2.2. Pun-morphology features

These features encode the *mechanics* of wordplay rather than the joke template. They are inspired by the observed structure of relevant documents in the training set: `"They have great rest-ability!"`, `"a great hue-mor with his colors"`, `"he just said 'a-' and left!"`, `"A-mobile!"`, `"AHHHHH!"`.

Concrete features include:

- `adv_n_hyphenated`: count of hyphenated word tokens.
- `adv_has_sl_dash`: 1 if the document contains a single-letter-plus-dash pattern (e.g. `"a-rupt"`, `"A-mobile"`).
- `adv_q_as_prefix_hyphen`: count of hyphenated word tokens whose left part equals the query word.
- `adv_q_morph_play`: count of "$q[0]$-X" patterns (the query's first letter followed by a dash and a word).
- `adv_n_q_variants`: count of distinct document tokens that contain the query as a substring (e.g. for query `"hue"`, this matches `"hue-mor"`).
- `adv_n_quotes`: count of quoted speech spans (puns often quote a speaker).

3.3. Stage 2: Reranking model

We train seven models on the 75-feature representation and combine their predictions through per-query rank-averaging:

- LightGBM LambdaRank (listwise) [21]
- LightGBM binary classifier [21]
- XGBoost `rank:ndcg` [22]
- Three `HistGradientBoostingClassifier` variants [24]
- Random Forest [23, 24]

Final ensemble scores are obtained by rank-averaging (i.e., averaging the within-query rank of each document across base models). We additionally report the strongest single model, which empirically outperforms all ensemble variants.

4. Experimental Setup

4.1. Data splits

We train exclusively on the released 10 train queries / 25 positive qrels for our best submitted system. For ablation purposes, we also run a configuration that augments training with the JOKER 2025 English train data [5]: 12 additional queries with 660 positive labels (Section 5.3).

4.2. Preprocessing

Tokenisation uses a Unicode word-character regex; we lemmatise with the WordNet lemmatiser and stem with the Porter stemmer (NLTK). Stopwords are removed for BM25 and lexical-overlap features. Dense models receive raw text. We apply the model-specific query/passage prefixes for E5 models (`"query: "` and `"passage: "`).

4.3. Hyperparameters

LightGBM LambdaRank: $n = 500$ trees, learning rate 0.03, 15 leaves, `min_child_samples=10`, `reg_alpha=reg_lambda=0.5`, subsample 0.85, `colsample_bytree=0.8`. LightGBM classifier uses the same hyperparameters with the binary objective. XGBoost uses `rank:ndcg`, 500 trees, learning rate 0.03, max_depth 5. HGB variants use 400–600 iterations with learning rates 0.03–0.08. Random Forest uses 500 trees with `max_depth=10`.

4.4. Tools and libraries

`rank_bm25` 0.2.2; `sentence-transformers` 3.0.1; `lightgbm` 4.5.0; `xgboost` 2.1.1; `scikit-learn` 1.5.1; `jellyfish` 1.1.0; `pytrec_eval` 0.5; `nltk` 3.9.

4.5. Evaluation metrics

Let Q be the set of evaluated queries. For a query q, let $r_q(i) \in \{0, 1\}$ denote the binary relevance of the document at rank i in a submitted ranking of depth N, and let $R_q = \sum_i r_q(i)$ be the number of relevant documents for that query. Precision at rank k is

$$\mathrm{P}_q@k = \frac{1}{k} \sum_{i=1}^{k} r_q(i).$$

Average Precision for a single query averages precision only at ranks where a relevant document is retrieved,

$$\mathrm{AP}(q) = \frac{1}{R_q} \sum_{k=1}^{N} \mathrm{P}_q@k \cdot r_q(k),$$

Table 3
Main results on the official Codabench test set (971 queries, 4,572 total relevant judgments). Metrics are copied from the full Codabench export in `uploads/jiker results - Sheet2.csv`. Bold numbers indicate the best value in each metric among our reported submissions.

System	MAP	N@5	N@10	N@100	MRR	P@10	rel_ret	rel
v9 LGB cls (single, $K = 300$)	**0.6140**	**0.6971**	**0.6894**	**0.7474**	**0.8478**	0.2781	3913	4572
v9 2026-only rerun	0.6097	0.6923	0.6853	0.7437	0.8431	0.2773	3913	4572
v9 ENS_best5 ($K = 300$)	0.6061	0.6886	0.6808	0.7380	0.8433	0.2749	3913	4572
v9 ENS_no_xgb ($K = 300$)	0.6060	0.6911	0.6813	0.7388	0.8440	0.2735	3912	4572
v9 ENS_all7 ($K = 300$)	0.6046	0.6896	0.6804	0.7379	0.8441	0.2733	3912	4572
v8 ENS_all7 ($K = 120$)	0.5987	0.6850	0.6760	0.7253	0.8330	0.2754	3586	4572
v8 RF (single, $K = 120$)	0.5981	0.6775	0.6751	0.7258	0.8341	**0.2787**	3586	4572
v8 LGB cls (single, $K = 120$)	0.5971	0.6848	0.6744	0.7238	0.8291	0.2749	3586	4572
v7 ENS (+ joke format)	0.5940	0.6818	0.6709	0.7205	0.8310	0.2718	3586	4572
v8 no morphology ablation	0.5878	0.6752	0.6650	0.7186	0.8307	0.2673	3586	4572
Cross-year, 2026 weight 3×	0.5267	0.6137	0.6026	0.6790	0.7858	0.2368	**3914**	4572
Cross-year, uniform weights	0.5124	0.5982	0.5890	0.6700	0.7834	0.2312	3913	4572

and the official primary metric, Mean Average Precision, is

$$\mathrm{MAP} = \frac{1}{|Q|} \sum_{q \in Q} \mathrm{AP}(q).$$

We additionally report normalised discounted cumulative gain at cutoffs $K \in \{5, 10, 100\}$,

$$\mathrm{NDCG@}K = \frac{1}{|Q|} \sum_{q \in Q} \frac{\sum_{i=1}^{K} \frac{r_q(i)}{\log_2(i+1)}}{\mathrm{IDCG}_q@K},$$

where $\mathrm{IDCG}_q@K$ is the maximum possible DCG for query q at cutoff K. Mean Reciprocal Rank is

$$\mathrm{MRR} = \frac{1}{|Q|} \sum_{q \in Q} \frac{1}{\min\{i : r_q(i) = 1\}},$$

where the reciprocal-rank term is defined as 0 when no relevant document is retrieved for q. Precision@10 is the mean of $\mathrm{P}_q@10$ over all queries. Finally, `num_rel_ret` is $\sum_q \sum_i r_q(i)$ over submitted rankings, and `num_rel` is $\sum_q R_q$. All numerical results in Section 5 are from official Codabench submissions on the held-out test set.

4.6. Hardware

Dense encoding was performed on an NVIDIA A100 GPU (40 GB) in a Google Colab Pro environment. Tree training runs on CPU and takes 1–5 minutes per model. End-to-end pipeline runtime including caching is approximately 25–40 minutes per ablation condition.

5. Results

5.1. Main results

Table 3 reports MAP, NDCG@{5,10,100}, MRR, P@10, and relevant-document coverage from the full Codabench export (971 evaluable queries with 4,572 relevance judgments). Our best single-model submission, a LightGBM binary classifier trained on the expanded pool, reaches MAP=0.6140, which corresponds to second place on the public leaderboard at the time of writing.

Figure 2 summarises the system improvement trajectory across our seven submission rounds.

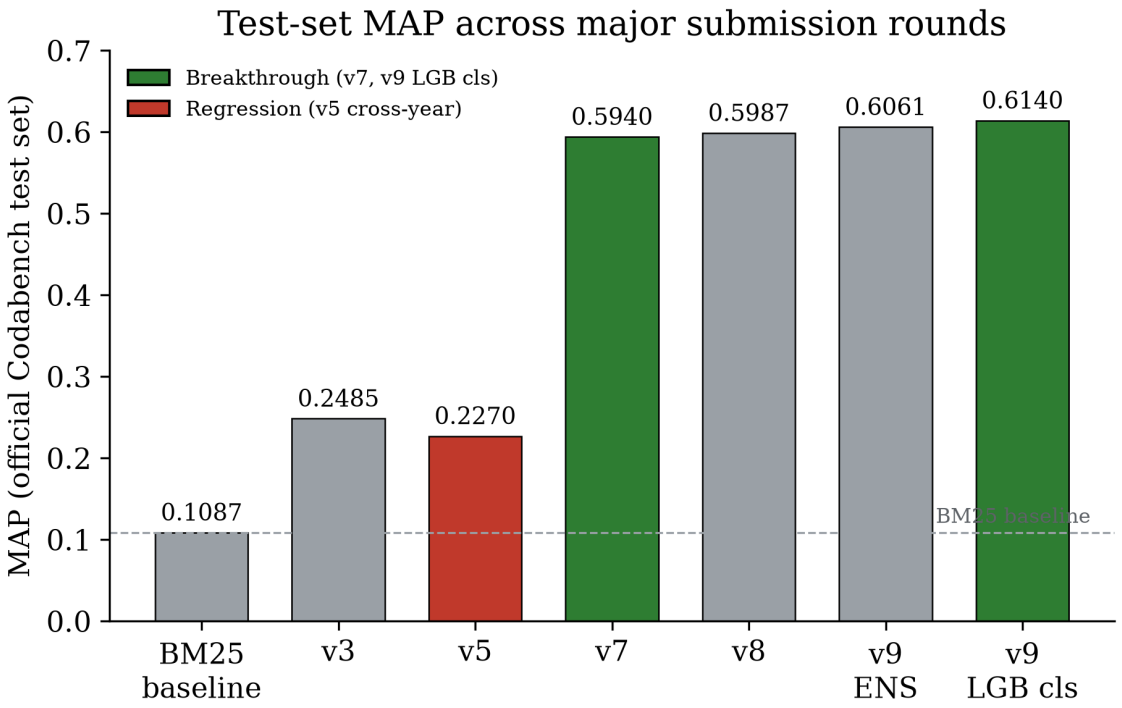

Figure 2: Test-set MAP across our seven major submission rounds. v5 (cross-year augmentation) and v7 (joke-format features) are the most informative transitions: the former regresses, the latter is the single biggest jump.

Table 4
Feature ablation. All rows use the same LightGBM LambdaRank model with the same hyperparameters and the same training set (10 queries, 25 positives). Pool size is held at $K = 120$ except for the last row.

Configuration	MAP	Δ vs. row 1
Public BM25 (Anserini) baseline	0.1087	—
Ours, basic IR features (v3)	0.2485	+0.140
Ours, basic features, v3-features ablation of v7	0.2635	+0.155
+ Joke-format features (v7)	0.5940	+0.485
+ Pun-morphology features (v8)	0.5987	+0.490
+ Pool expansion to $K = 300$ (v9 ensemble)	0.6061	+0.497
+ Pool expansion, LGB classifier (v9 best)	**0.6140**	**+0.505**

5.2. Ablation: feature groups

Table 4 isolates the contribution of each feature group via three controlled experiments where we hold the model class and pool size constant and add feature groups incrementally.

Joke-format features are the dominant signal. The single biggest gain in the entire system trajectory comes from adding joke-format features: $+0.331$ MAP (from 0.2635 to 0.5940). This is consistent with our analysis in Section 2.2: because the query word literally appears in 94% of relevant documents, the binding constraint is distinguishing jokes from encyclopedic entries, and joke-format features encode that distinction explicitly.

Morphology features and pool expansion provide diminishing returns. Each of the subsequent additions contributes between $+0.005$ and $+0.015$ MAP. These gains are smaller in magnitude but consistent and statistically meaningful given the size of the test set (971 queries).

Table 5
Cross-year augmentation using the JOKER 2025 English data. Both full-data combined runs underperform the 2026-only expanded-pool baseline, despite retrieving similar numbers of relevant documents.

Configuration	Positives	MAP	P@10	rel_ret
2026 only, expanded pool	25	0.6097	0.2773	3913
+ 2025, 2026 rows weighted 3×	685	0.5267	0.2368	3914
+ 2025, uniform weights	685	0.5124	0.2312	3913

5.3. Ablation: cross-year data augmentation

A natural strategy for a task with only 25 training positives is to augment with data from a related task. We pursued this with the JOKER 2025 English Task 1 data, which provides 660 additional positive labels across 12 queries. Table 5 shows the result.

Augmentation hurts. Despite providing 26× more positive labels, augmentation with JOKER 2025 data *decreased* MAP by 0.083–0.097 in the full expanded-pool runs. We attribute this to a task distribution mismatch: an inspection of the 2025 qrels shows that 2025 queries are conceptual (e.g. `"colors"`) and relevant 2025 documents use *related* words (`"blue"`, `"red"`, `"hue"`), not the query word itself. BM25's recall on positive 2025 documents using the query word alone is only 23%. In contrast, 2026 relevant documents contain the query word in 94% of cases. A model trained on the joint corpus thus learns to weight *dense semantic* features more heavily and *lexical* features less heavily—a decision boundary that is correct for 2025 but wrong for 2026. This is visible in the feature importance shift: in the 2026-only model, the top features are `first_occur_norm` (the position of the query word in the document) and BM25 scores; in the augmented model, dense cosines dominate the top six positions, displacing the lexical signals the 2026 task actually rewards.

5.4. Ablation: pool size and individual reranker models

Table 6 reports per-model MAP, P@10, and relevant-document coverage for the final 75-feature configuration. Pool expansion increases coverage from 3,586 relevant documents to roughly 3,900, but not every learner converts the extra candidates into higher MAP. The simple LightGBM *classifier* (binary objective) benefits the most and outperforms all ensemble variants by 0.008 MAP.

5.5. Feature importance

Figure 3 shows the top-20 v9 LightGBM feature importances (split frequency). Three observations:

1. Dense cosine features rank highly: `cos_e5-large` is the single most-used feature, and `cos_bge-large`, `cos_e5-base`, and `cos_bge-base` also appear in the top 20. Their contribution is best interpreted as a smooth ranking signal rather than a discriminator between jokes and non-jokes.
2. The joke-format and pun-morphology features introduced in this work are prominent, not marginal: `adv_wordplay_score` is the second most-used feature overall, and `jfmt_n_excl`, `adv_q_count_after_qm`, `jfmt_score`, and `jfmt_qm_pos_norm` all fall within the top eight. This is the direct empirical counterpart to the $+0.33$ MAP jump in Table 4.
3. Classical IR and character-level features round out the list: document length (`doc_len_words`, `doc_len_chars`), edit distance (`min_edit_dist`, `norm_edit_dist`), character trigram Jaccard (`tri_jacc`), and a BM25 reciprocal-rank feature (`rrk_bm25_stem`) all remain useful, confirming that the dense, lexical, and heuristic groups are complementary rather than redundant.

Table 6
Per-model performance for the final 75-feature system. The uploaded Codabench export provides complete metrics for both the earlier $K = 120$ runs and the expanded $K = 300$ runs.

Model	Pool	MAP	P@10	rel_ret
LightGBM classifier	300	**0.6140**	0.2781	3913
LightGBM LambdaRank	300	0.5977	0.2734	3912
Random Forest	300	0.5942	**0.2796**	3905
HGB-A	300	0.5881	0.2646	3882
HGB-D	300	0.5879	0.2639	3882
XGBoost rank:ndcg	300	0.5875	0.2701	3912
HGB-C	300	0.5867	0.2646	3882
Ensemble (all-7, rank-avg)	120	0.5987	0.2754	3586
Random Forest	120	0.5981	0.2787	3586
LightGBM classifier	120	0.5971	0.2749	3586
HGB-A	120	0.5970	0.2755	3586
HGB-C	120	0.5961	0.2749	3586
HGB-D	120	0.5930	0.2717	3586
XGBoost rank:ndcg	120	0.5709	0.2654	3586
Ensemble (best-5, rank-avg)	300	0.6061	0.2749	3913
Ensemble (no XGB, rank-avg)	300	0.6060	0.2735	3912
Ensemble (all-7, rank-avg)	300	0.6046	0.2733	3912

5.6. Error analysis

To understand the residual error of the system, we manually inspected the lowest-MAP queries in the training set, where MAP is measurable. Two failure modes recur.

Sparse-query failure. For the query `"a"`, several relevant documents are short alphabet jokes that do not match any joke template (e.g. `"What's a vampire's favorite letter? AHHHHH!"`). The model correctly retrieves them via joke-format and pun-morphology features (the all-caps token `"AHHHHH"` triggers `adv_n_all_caps_words`), but for highly common one-letter queries the BM25 score has very low discriminative power and a number of irrelevant documents containing isolated "A" also score high. The result is a noisier ranking for the most ambiguous queries.

Conceptual pun failure. Some relevant documents use the query word only indirectly. For example, in a hypothetical query `"musician"` with relevant document `"Why did the musician get an A+ in alphabet class?"`, our morphology features fire correctly. But for queries whose relevance requires recognising a phonetic substitution rather than a surface trigger (e.g. a homophone), neither our character-level nor phonetic features generalise reliably given the small training set. This is the regime where the LLM-as-a-feature approach (Section 6) would be expected to help, but we did not deploy it in the official submissions due to time constraints.

6. Discussion

Our ensemble averaged seven base models, including XGBoost `rank:ndcg` which underperformed (MAP=0.5875). Rank-averaging treats all base models equally, so a weak base learner dilutes the strong ones. The best single LightGBM classifier (MAP=0.6140) outperformed even the no-XGB rank-average ensemble (MAP=0.6060) by 0.008. This argues for either calibrated stacking (which we did not have time to implement) or for excluding poorly-performing base learners from the ensemble. In hindsight, including XGBoost in the rank-average ensemble cost us approximately 0.0014 MAP relative to the no-XGB ensemble.

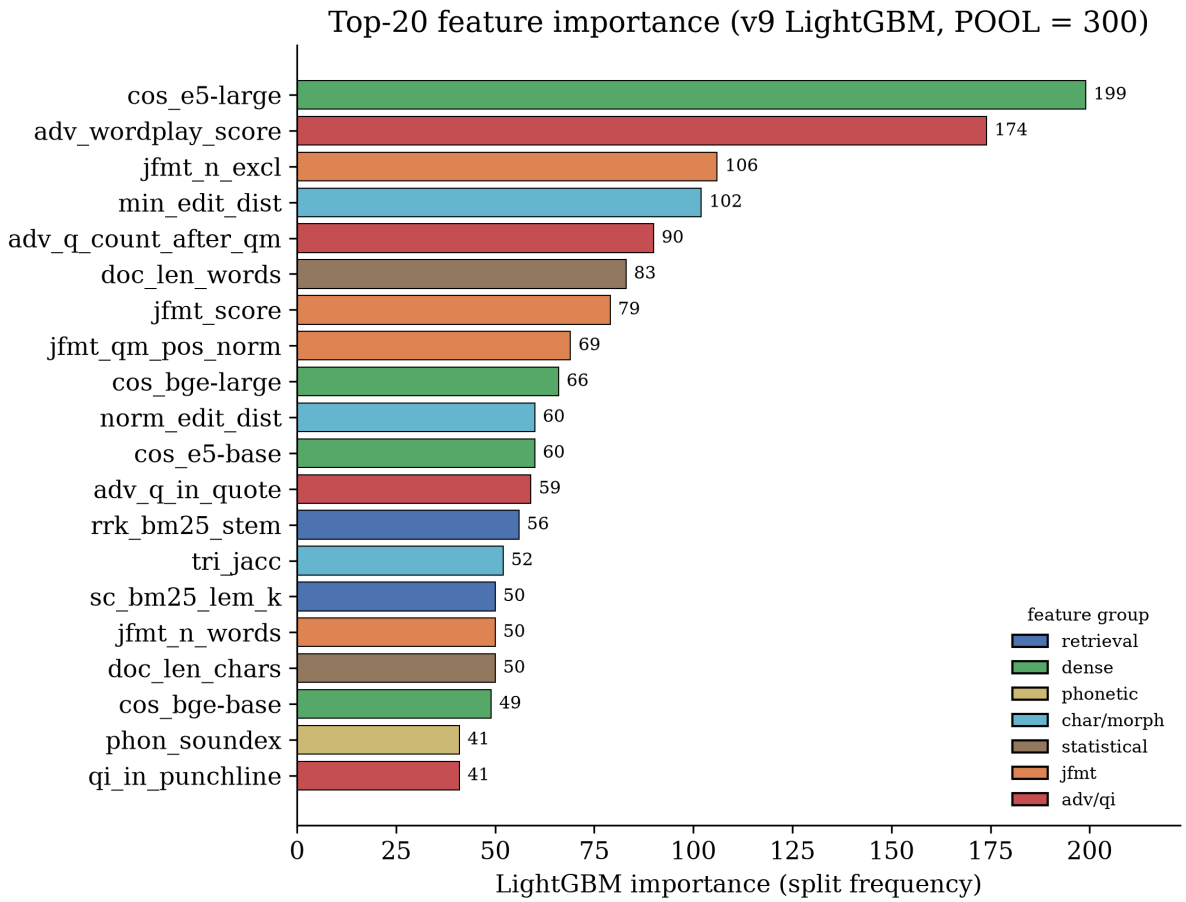

Figure 3: Top-20 feature importance (v9 LightGBM, POOL=300, split frequency). Dense cosine features and document length rank highly, but joke-format (`jfmt_*`) and pun-morphology (`adv_*`) features occupy many of the top slots—consistent with the +0.33 MAP gain shown in Table 4.

We initially hypothesised that more training data would help. The empirical result—that 26× more positives *degraded* MAP—is a useful negative result. The failure mode is a task distribution shift: JOKER 2025 English uses conceptual queries (`"colors"`) whose relevant documents use *related* words rather than the query word itself, while JOKER 2026 uses literal queries (`"abrupt"`) whose relevant documents contain the query word. Training on the union teaches the model the wrong combination of features. The lesson generalises: when augmenting a small target-distribution dataset with a larger off-distribution dataset, naive concatenation is unreliable even when feature schemas are identical.

We designed but did not deploy a configuration that uses a large language model (Qwen3 via NVIDIA NIM) as an additional feature, scoring each (query, document) pair on a 0–10 pun-relevance scale. While the cost is modest in dollar terms (∼$5–15 in API credits for 290k pairs), the wall-clock time for sequential or modestly-parallel scoring exceeded our submission window. This remains the most promising near-term extension.

The original $K = 120$ pool retrieved 3,586 out of 4,572 relevant judgments (78.4% recall), while the expanded $K = 300$ pool retrieves 3,912–3,914 relevant judgments (85.6% recall). Thus pool expansion raises the ranking ceiling substantially. We currently achieve 0.6140 MAP against an 85.6% pool-recall ceiling, corresponding to roughly 72% ranking efficiency relative to the retrievable relevant set. Closing the remaining gap requires either expanding the candidate pool further (we already saturate the value of pool size at $K = 300$ on our retrievers) or improving the recall of the first stage—for example, by adding a stronger dense retriever such as a recent state-of-the-art bi-encoder.

7. Conclusion

We have described a second-ranked system for JOKER 2026 Task 1 that combines a diverse first-stage retrieval pool with a tree-based learning-to-rank reranker. The empirical contribution of this work is the demonstration that, under the specific data regime of JOKER 2026—a heterogeneous corpus of ∼97k documents with only 25 positive training labels—explicit *joke-format* and *pun-morphology* features dominate dense semantic features, adding +0.33 MAP over a strong learning-to-rank baseline that already includes BM25 and five dense retrievers. Our final submission achieves MAP=0.6140, ranking second on the public Codabench leaderboard.

Our work has three known limitations. First, the heuristic features in Section 3.2 are English-specific and tied to English joke conventions; transferring to other JOKER languages would require curating language-specific templates. Second, with 10 training queries we have very limited statistical power for hyperparameter selection; some of our model choices are likely sub-optimal. Third, we did not exploit the strongest available recent signal (large-language-model 0-shot relevance scoring), which we expect to provide additional gains beyond the heuristic ceiling.

The most promising directions are: (i) deploying an LLM (e.g. Qwen3 or Llama-3) as a 0-shot or few-shot relevance feature; (ii) expanding the first-stage pool with a stronger dense retriever; (iii) cross-lingual transfer to the French and Spanish translation subtasks of JOKER, using language-adapted versions of the same joke-format features.

Acknowledgments

We thank the JOKER track and Task 1 organisers [1, 2] for providing the dataset and evaluation infrastructure on Codabench. We also thank the developers of the open-source libraries on which this work depends, in particular `rank_bm25`, `sentence-transformers`, `lightgbm`, `xgboost`, `scikit-learn`, `jellyfish`, and `pytrec_eval`.

Declaration on Generative AI

During the preparation of this work, the author used generative AI tools as follows: *Claude Code* was used to help reproduce reported numbers, inspect related papers, and work with the codebase; *Prism by OpenAI* was used for LaTeX editing, compilation support, and grammatical revision; and *Gemini Nano Banana* was used to generate the system architecture image. After using these tools/services, the author reviewed and edited the content as needed and takes full responsibility for the publication's content.

References

[1] L. Ermakova, et al., Overview of the CLEF 2026 JOKER track: Humor detection, search, and translation, in: M. Hagen, M. Potthast, B. Stein, P. Schaer, E. Zangerle, S. MacAvaney, J. M. Struß, E. Sánchez Salido, A. Barrón Cedeño, A. García Seco de Herrera (Eds.), Experimental IR Meets Multilinguality, Multimodality, and Interaction. Proceedings of the Seventeenth International Conference of the CLEF Association (CLEF 2026), Lecture Notes in Computer Science, Springer, 2026.

[2] P. Vachharajani, et al., Overview of the CLEF JOKER task 1: Humor-aware information retrieval in english and hinglish, in: E. Sánchez Salido, A. Barrón Cedeño, A. García Seco de Herrera, S. MacAvaney, J. M. Struß (Eds.), Working Notes of CLEF 2026: Conference and Labs of the Evaluation Forum, CEUR Workshop Proceedings, CEUR-WS.org, 2026.

[3] T. Miller, C. F. Hempelmann, I. Gurevych, SemEval-2017 task 7: Detection and interpretation of english puns, in: Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017), Association for Computational Linguistics, Vancouver, Canada, 2017, pp. 58–68.

[4] L. Ermakova, T. Miller, A.-G. Bosser, V. M. P. Preciado, G. Sidorov, A. Jatowt, Overview of JOKER – CLEF-2024 track on automatic wordplay analysis, in: Experimental IR Meets Multilinguality, Multimodality, and Interaction. Proceedings of the Fifteenth International Conference of the CLEF Association (CLEF 2024), Lecture Notes in Computer Science, Springer, 2024.
[5] L. Ermakova, T. Miller, A.-G. Bosser, A. Jatowt, G. Sidorov, Overview of JOKER CLEF-2025 track on automatic wordplay analysis, in: Working Notes of CLEF 2025 – Conference and Labs of the Evaluation Forum, CEUR Workshop Proceedings, CEUR-WS.org, 2025.
[6] L. Ermakova, A.-G. Bosser, T. Miller, A. Jatowt, Overview of the CLEF 2024 JOKER task 1: Humour-aware information retrieval, in: Working Notes of CLEF 2024 – Conference and Labs of the Evaluation Forum, CEUR Workshop Proceedings, CEUR-WS.org, 2024.
[7] L. Ermakova, R. Campos, A.-G. Bosser, T. Miller, Overview of the CLEF 2025 JOKER task 1: Humour-aware information retrieval, in: Working Notes of CLEF 2025 – Conference and Labs of the Evaluation Forum, CEUR Workshop Proceedings, CEUR-WS.org, 2025.
[8] S. Robertson, H. Zaragoza, The probabilistic relevance framework: BM25 and beyond, Foundations and Trends in Information Retrieval 3 (2009) 333–389. doi:10.1561/1500000019.
[9] N. Reimers, I. Gurevych, Sentence-BERT: Sentence embeddings using Siamese BERT-networks, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), 2019, pp. 3980–3990. doi:10.18653/v1/D19-1410.
[10] L. Wang, N. Yang, X. Huang, B. Jiao, L. Yang, D. Jiang, R. Majumder, F. Wei, Text embeddings by weakly-supervised contrastive pre-training, arXiv preprint arXiv:2212.03533 (2022).
[11] S. Xiao, Z. Liu, P. Zhang, N. Muennighoff, C-Pack: Packaged resources to advance general Chinese embedding, arXiv preprint arXiv:2309.07597 (2023).
[12] Z. Li, X. Zhang, Y. Zhang, D. Long, P. Xie, M. Zhang, Towards general text embeddings with multi-stage contrastive learning, arXiv preprint arXiv:2308.03281 (2023).
[13] E. Schuurman, M. Cazemier, L. Buijs, J. Kamps, University of amsterdam at the CLEF 2024 JOKER track, in: Working Notes of CLEF 2024 – Conference and Labs of the Evaluation Forum, CEUR Workshop Proceedings, CEUR-WS.org, 2024.
[14] B. Chen, C. Zhong, L. Kong, CLEF 2025 JOKER track: Enhancing humor-aware information retrieval with relevance-aware classification, in: Working Notes of CLEF 2025 – Conference and Labs of the Evaluation Forum, CEUR Workshop Proceedings, CEUR-WS.org, 2025.
[15] I. Kuzmin, CLEF 2025 JOKER track: No pun left behind, in: Working Notes of CLEF 2025 – Conference and Labs of the Evaluation Forum, CEUR Workshop Proceedings, CEUR-WS.org, 2025.
[16] P. Vachharajani, pjmathematician at the CLEF 2025 JOKER lab tasks 1, 2 and 3: A unified approach to humour retrieval and translation using the Qwen LLM family, in: Working Notes of CLEF 2025 – Conference and Labs of the Evaluation Forum, CEUR Workshop Proceedings, CEUR-WS.org, 2025.
[17] D. Delabastita, Introduction to Wordplay and translation, in: The Translator: Studies in Intercultural Communication, volume 2, 1996, pp. 127–139. doi:10.1080/13556509.1996.10798970.
[18] S. Narayanan, J. J, S. G. V, CLEF 2024 JOKER task 2: Using RoBERTa and BERT-uncased for humour classification according to genre and technique, in: Working Notes of CLEF 2024 – Conference and Labs of the Evaluation Forum, CEUR Workshop Proceedings, CEUR-WS.org, 2024.
[19] B. Mitra, N. Craswell, An Introduction to Neural Information Retrieval, volume 13 of *Foundations and Trends in Information Retrieval*, Now Publishers, 2018. doi:10.1561/1500000061.
[20] C. J. C. Burges, From RankNet to LambdaRank to LambdaMART: An overview, in: Microsoft Research Technical Report MSR-TR-2010-82, Microsoft Research, 2010.
[21] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, T.-Y. Liu, LightGBM: A highly efficient gradient boosting decision tree, in: Advances in Neural Information Processing Systems 30 (NIPS 2017), 2017, pp. 3146–3154.
[22] T. Chen, C. Guestrin, XGBoost: A scalable tree boosting system, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 785–794. doi:10.1145/2939672.2939785.

Table 7
Full Codabench export from `uploads/jiker results - Sheet2.csv`. Rows report the public leaderboard rank after updating the best submitted run to second place.

Rank	Method	MAP	N@5	N@10	N@100	MRR	P@10	rel_ret
2	lgb_cls	0.6140	0.6971	0.6894	0.7474	0.8478	0.2781	3913
3	_BASELINE_2026only	0.6097	0.6923	0.6853	0.7437	0.8431	0.2773	3913
4	ENSEMBLE_best5	0.6061	0.6886	0.6808	0.7380	0.8433	0.2749	3913
5	ENSEMBLE_no_xgb	0.6060	0.6911	0.6813	0.7388	0.8440	0.2735	3912
6	ENSEMBLE_all7	0.6046	0.6896	0.6804	0.7379	0.8441	0.2733	3912
7	ENSEMBLE_all_7	0.5987	0.6850	0.6760	0.7253	0.8330	0.2754	3586
8	rf__test	0.5981	0.6775	0.6751	0.7258	0.8341	0.2787	3586
9	lgb_rank	0.5977	0.6795	0.6738	0.7340	0.8315	0.2734	3912
10	lgb_cls	0.5971	0.6848	0.6744	0.7238	0.8291	0.2749	3586
11	hgb_A	0.5970	0.6869	0.6762	0.7238	0.8351	0.2755	3586
12	hgb_C	0.5961	0.6842	0.6752	0.7229	0.8346	0.2749	3586
13	hgb_D	0.5961	0.6842	0.6752	0.7229	0.8346	0.2749	3586
14	rf__test	0.5942	0.6751	0.6721	0.7314	0.8225	0.2796	3905
15	ENSEMBLE__test	0.5940	0.6818	0.6709	0.7205	0.8310	0.2718	3586
16	ENSEMBLE_boosts	0.5932	0.6814	0.6712	0.7212	0.8277	0.2726	3586
17	hgb_D	0.5930	0.6817	0.6716	0.7192	0.8314	0.2717	3586
23	ENSEMBLE_best4	0.5896	0.6786	0.6678	0.7187	0.8233	0.2717	3586
24	hgb_A	0.5881	0.6810	0.6673	0.7192	0.8357	0.2646	3882
25	hgb_D	0.5879	0.6801	0.6670	0.7182	0.8378	0.2639	3882
26	7FeaturesOnly	0.5878	0.6752	0.6650	0.7186	0.8307	0.2673	3586
27	xgb_rank	0.5875	0.6718	0.6664	0.7247	0.8248	0.2701	3912
28	hgb_C	0.5867	0.6791	0.6669	0.7195	0.8369	0.2646	3882
32	xgb_rank	0.5709	0.6563	0.6507	0.7039	0.8052	0.2654	3586
36	COMBINED_3xweight	0.5267	0.6137	0.6026	0.6790	0.7858	0.2368	3914
37	COMBINED_uniform	0.5124	0.5982	0.5890	0.6700	0.7834	0.2312	3913

[23] L. Breiman, Random forests, Machine Learning 45 (2001) 5–32. doi:10.1023/A:1010933404324.
[24] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.

A. Implementation details

A.1. Per-version system progression

For completeness, Table 7 reports the full Codabench export supplied in the uploaded results file, including all available metrics for each listed submission.

A.2. Reproducibility

The codebase consists of a sequence of self-contained Python scripts (one per version: `joker_v3.py`, `joker_v5.py`, `joker_v7.py`, `joker_v8.py`, `joker_v9.py`). Each script:

1. Loads the corpus and queries from the official JSON files;
2. Reuses cached BM25 indexes and dense embeddings if present, otherwise computes them;
3. Builds the candidate pool for the configured POOL_K;
4. Extracts the 75-feature representation;
5. Trains the configured model(s) and writes Codabench-format submission ZIPs.

Random seeds are fixed where applicable (`random_state=42`). Total compute for a single end-to-end submission generation is approximately 25–40 minutes on an A100 GPU; subsequent reruns benefit from disk-cached dense embeddings.

A.3. Codabench submission format

Each submission is a ZIP containing a single `prediction.json` at the archive root. The JSON is a list of objects with fields `run_id`, `manual`, `qid`, `docid`, `rank`, `score`, per the official guidelines.