

Transforming Data Overload into Value: AI-Based Enhancements to the Information Gain Methodology

Tanushree Gupta

San Diego, California, USA

Email: [tanushree.851\[at\]gmail.com](mailto:tanushree.851[at]gmail.com)

Abstract: Organizations keep far more data than they use, and low-value records raise storage cost and cause information overload. This paper asks how to decide automatically whether a new record belongs in the primary database or in an archive, without deleting data. We correct an earlier entropy-based information gain (IG) method that counted keywords, redefine IG as the Kullback–Leibler divergence between a record’s topic distribution and the corpus distribution, and add relevance, novelty, and supervised scoring with SHAP explanations. A simulation with 30 seeds of a synthetic corpus and 20 seeds of real PyPI package descriptions compared the methods at a 95% recall target, with paired significance tests. Keyword, TF-IDF, and BM25 scoring had to retain every synthetic record. Semantic IG with relevance reached F1 0.860 against 0.439 and archived 65% of records. On real text the gain was small: F1 0.418 against 0.397. A MinHash duplicate filter outperformed embedding novelty for near-exact copies, and neither handled heavily edited copies. The encoder is latent semantic analysis, so pretrained encoders remain untested.

Keywords: Information gain; Entropy; Data retention; Sentence embeddings; Explainable AI

1. Introduction

Enterprise databases grow every day with both facts and records of how those facts were transmitted. Organizations that rely on knowledge management keep spending on storage, maintenance, and processing, including information mining [1]. More data does not mean more information. Redundant or irrelevant records raise storage and maintenance cost, slow queries, and bury the records that matter. Regulation limits what can be dropped. The Information Technology Act, 2000 in India and the GDPR in the EU both constrain retention, so deletion is often not an option.

A module is needed to test whether a new entry adds to the worth of the database. We use information gain for this. It measures the information content of an answer given the prior probabilities of getting the relevant answer to a question [2,3]. Here its job is to help the user judge whether the entered data is useful.

The first module [4,5], built by two students from Maharaja Agrasen Institute of Technology, New Delhi, during an internship at the DIT, scored each incoming record with an entropy-based measure over predefined keywords. Accepted records went to a primary database, which is used for result computation. The rest went to a secondary database that keeps them as history. The design has gaps. Relevance came from a fixed keyword list. The gain was computed on keyword counts and ignored meaning. Redundancy was never measured. The decision rule, the slope of a plotted graph, was loosely specified. And nothing was evaluated. This paper addresses each gap.

The work has five parts. It gives the corrected entropy and conditional-entropy formulas, worked examples, and reference code. It defines record-level information gain over embedding-derived topic distributions, adds a novelty term

for redundant records, and tests a multiplicative relevant-and-novel rule. It adds supervised relevance prediction with SHAP explanations and a contextual-bandit policy, and sets out a retain, archive, and expire lifecycle with legal holds and an audit trail. It gives an evaluation protocol with paired significance tests and MinHash and SimHash baselines. And it reports the results of running that protocol on a synthetic corpus and on real package descriptions.

2. Literature Survey

The idea of measuring information by the uncertainty it removes goes back to Shannon [6,7]. Decision-tree induction later made information gain a practical splitting criterion [2,8], and text categorization adopted it for term selection [9]. The strands below bear on record retention.

Information gain and feature selection: IG comes from information theory [6,7] and decision-tree induction [2,8]. In text categorization it is a standard way to pick informative terms, and it has been compared with document frequency, mutual information, and χ^2 [9]. We score records for retention, not terms for classification. IG-based analysis has also been used to explain agent behaviour in trading simulations [10].

Term weighting and retrieval: TF-IDF [11] and BM25 [12] weight terms by how distinctive they are. They serve as the lexical baselines for relevance. LDA [13] and embedding-based BERTopic [14] give unsupervised topic distributions, which can act as the event space for entropy.

Redundancy and near-duplicate detection: MinHash over shingles [15] and SimHash [16] find near-duplicate documents at scale. They flag redundancy but say nothing about topical value.

Surprise and novelty: Bayesian surprise is the KL divergence between posterior and prior beliefs, proposed as a measure of how much an observation changes a model [17]. KL-divergence kernels have been used to classify multimedia data [18]. Our record-level IG follows the surprise view, with the corpus topic distribution as the prior.

Semantic representations: Transformer encoders [19] and Siamese sentence encoders [20] produce embeddings whose cosine similarity tracks meaning. XLM-R [21] extends this to many languages. CLIP [22,23] aligns images with text, and vision transformers [24] encode images. FAISS [25] and hosted vector databases [26] provide nearest-neighbour search over the embeddings.

Feedback, explanation, and privacy: Contextual bandits [27] learn policies from partial feedback. SHAP [28] attributes a prediction to input features. Differential privacy [29] and federated learning [30] limit what leaks from the data. Retrieval-augmented generation [31] is one reason to keep low-priority data searchable instead of deleting it.

Data lifecycle: Information lifecycle management tiers data by value and age. The primary and secondary stores here are a two-tier case, with the tier chosen by content.

None of these strands addresses record-level retention. Feature selection scores terms, deduplication ignores value, and lifecycle management tiers by age. A content-aware retention score with an explicit recall target is the gap, and it is the problem defined next.

3. Problem Definition

Records $x_1, x_2, \dots$ arrive at an organization as a stream. Each record has text, sometimes an image, and metadata. The organization keeps a primary store D that supports result computation, and a secondary store that holds everything else. For each arriving record the system chooses to retain it, in which case it joins D , or to archive it. A record is *valuable* if it falls within the organization's mandate and adds information that D lacks, that is, it is not a near-duplicate of a stored record.

The goal is to retain the valuable records while archiving as many other records as possible. For a target recall ρ on valuable records, the decision rule $d(x)$ must satisfy *maximize share of records archived subject to recall(valuable records retained) $\geq \rho$.*

Archiving is not deletion. A model score is never the basis for removing data, because retention law (the Information Technology Act, 2000 and the GDPR) can require the data to be kept.

Version 1.0 of the method judged relevance with a fixed keyword list, ignored redundancy, and used the sign of a plotted slope as its decision rule. Four questions follow. **RQ1:** does semantic IG with a relevance term beat lexical scoring at the same recall target? **RQ2:** does a novelty term reduce redundancy in the primary store? **RQ3:** does a lexical duplicate filter such as MinHash make the embedding novelty

term unnecessary? **RQ4:** which signals carry the decision? Section 5 answers them.

4. Methodology/ Approach

4.1 Information Gain and Data Integration

The unit of information is the bit. A single binary answer carries $-\log_2 P(v)$ bits, where $P(v)$ is the probability of event v [6,7]. Table I shows funds released for projects and funds spent over the first two quarters of a financial year.

Table I: Funds released and spent for two quarters of the financial year (the Released/Spent rows of each project are shown on one line).

Sr. No	Project	Funds	Status
1	Hardware Manufacturing	Released / Spent	Yes/ No
2	Research & Development	Released / Spent	Yes/ Yes
3	Security	Released / Spent	No/ No
4	Delivery of services	Released / Spent	Yes/ Yes
5	Infrastructure Development	Released / Spent	Yes/ Yes

The training set covers two quarters of financial performance of the budget division in the corporate office. Funds were released for 80% of the projects and spent for 60%. The question is how much information is gained by testing whether funds were released or spent. For p positive and n negative instances the information content, or entropy, is $I(p/(p+n), n/(p+n)) = -(p/(p+n)) \log_2(p/(p+n)) - (n/(p+n)) \log_2(n/(p+n))$,

where $I(\cdot)$ is the entropy, $p/(p+n)$ the probability of the positive event, $\log_2(p/(p+n))$ its information content, $n/(p+n)$ the probability of the negative event, and $\log_2(n/(p+n))$ its information content [6,7,11]. By convention $0 \cdot \log_2 0 = 0$. Next we measure the information still needed after the test. An attribute A with v distinct values splits the dataset into v subsets, one per value. The information still needed after testing A is the conditional entropy $H(Y|A) = \sum_i [(p_i + n_i)/(p + n)] \cdot I(p_i/(p_i + n_i), n_i/(p_i + n_i))$ (conditional entropy),

The difference between the entropy and the conditional entropy is the information gain [2,8]: $IG(A) = I(p/(p+n), n/(p+n)) - H(Y|A)$.

$IG(A)$ is never negative. It is zero exactly when A is statistically independent of the outcome in the sample. An attribute therefore cannot have a negative gain. When this paper speaks of a negative change, it means a trend across successive entries (Section 4.3(C)), not IG itself.

A. Worked example

Take each of the ten rows of Table I as an instance whose outcome Status is Yes (positive) or No (negative). The attribute tested is Funds, with values Released and Spent. Seven rows are Yes and three are No, so the entropy is $I(0.7, 0.3) = 0.8813$ bits. Split by Funds, Released gives $(p,n) = (4,1)$ with $I = 0.7219$, and Spent gives $(3,2)$ with $I = 0.9710$. Then $H(Y|Funds) = \frac{1}{2}(0.7219) + \frac{1}{2}(0.9710) = 0.8464$, $IG(Funds) = 0.8813 - 0.8464 = 0.0349$ bits.

Knowing whether the row refers to released or spent funds therefore tells us very little (0.035 bits) about the Status outcome.

B. Sample problem

The implementation (Section 4.4) uses this sample. The input string is “The cyber policy needs to be reformed to reduce the incidents of cyber attacks and cyber terrorism.” and the keywords are {policy, incidents, cyber}. The sentence has 17 tokens, and five of them are keyword occurrences (cyber three times, policy, incidents). The preprocessor reports $p = 5$ and $n = 12$. The per-keyword counts are $pkey1 = 1$, $nkey1 = 4$; $pkey2 = 3$, $nkey2 = 2$; $pkey3 = 1$, $nkey3 = 4$. The implementation reports a gain of 0.1816255 (Fig. 6).

The three keyword subsets hold 5, 5, and 5 instances, 15 in total, so the conditional-entropy weights are taken over those 15 instances ($p = 5$, $n = 10$). The computation is $I(5/15, 10/15) = 0.9183$; $H(Y|keyword) = (5/15)(0.7219 + 0.9710 + 0.7219) = 0.8049$; $IG = 0.9183 - 0.8049 = 0.1134$ bits.

C. A record-retention example

The formula is more useful for measuring how well a keyword predicts the retention decision. Suppose analysts labelled twelve past records *retain* (positive) or *archive* (negative), six of each, so the entropy is 1 bit. The keyword “cyber” appears in six records, and five of them were retained. It is absent from the other six, one of which was retained. Then $I(5/6, 1/6) = 0.6500$ for both subsets, so $H(Y|cyber) = 0.6500$ and $IG(cyber) = 0.3500$ bits. Had the keyword appeared equally often in both classes (3,3 and 3,3), the gain would be 0, and the keyword would say nothing about retention. The keyword baseline in Section 4.7 is trained this way.

4.2 Vector Representation for Non-Numerical Data

Some data sets suit a vector representation better than the scalar approach above. Two vectors x and y are orthogonal when their inner product is zero, meaning they are perpendicular. The numerical vectors $(1, 3, 2)$, $(3, -1, 0)$, and $(1/3, 1, -5/3)$ are orthogonal to one another. The pairwise dot products are $1 \cdot 3 - 3 + 0 = 0$, $1/3 + 3 - 10/3 = 0$, and $1 - 1 + 0 = 0$. Corporate data is largely non-numerical and stored as strings, so orthogonality has to be computed on strings.

A. Orthogonality of functions

Two functions f and g are orthogonal on $[a, b]$ if $\int_a^b f(x)g(x)dx = 0$ [32]. This generalizes orthogonality of vectors.

B. Orthogonality of strings

The concept is extended to check the data being entered. If the entered attributes are orthogonal, the entry does not add to the information gain of the system and is a potential source of overload. A query is treated as a vector whose components are the instruction fields. One field names the information to search for, and another gives the input information. Table II illustrates why strings need an orthogonality check, using the objective “Expansion & Participation” from the DIT RFD [33].

Table II: Orthogonality of strings (RFD of the Department of Information Technology)

No.	Objective	Action	Success indicator
1	Expansion & Participation	Setting up of 10 new universities	(i) Allocation of land; (ii) Appointment of Director; (iii) Prospectus circulation in banks; (iv) Admissions for courses

Represent each text as a vector, either TF-IDF counts or an embedding. The cosine similarity $\cos(x, y) = x \cdot y / (|x||y|)$ is zero when the vectors are orthogonal, that is, when the texts share no weighted vocabulary. The four success indicators in Table II (“Allocation of land,” “Appointment of Director,” “Prospectus circulation in banks,” “Admissions for courses”) share almost no vocabulary. They are nearly orthogonal, so each one is a separate piece of information about the objective and none is redundant. A record orthogonal to everything stored is evidence of **novelty**. A cosine near 1 signals redundancy. Take a vocabulary (cyber, policy, reform, campus). The records $a = (1, 1, 1, 0)$ and $b = (1, 1, 0, 0)$ have $\cos = 2/(\sqrt{3} \cdot \sqrt{2}) = 0.816$ and largely repeat each other. The record $c = (0, 0, 0, 1)$ has $\cos(a, c) = 0$ and is entirely new. Novelty is not value, though, so two signals are kept apart: **relevance**, whether the record is on mandate, and **novelty**, whether it adds anything the corpus lacks. Section 4.5 combines them.

4.3 Information Gain Methodology

The information-gain method was applied to the Results Framework Document (RFD) [33,34]. The RFD is voluminous text, so the effect of adding or deleting data on the information gain has to be assessed. When a user enters data into the database of the Department of Information Technology (DIT), the module shows the information gain status as a number and as a graph.

A. Steps

The procedure has six steps. First, the user enters the data to be verified by the information gain module. Second, the user selects the category and type of data. A category covers the divisional data, which can be strings or JPEG images [34].

Third, the module reads the predefined keywords. They are department specific and chosen by the module developers. For DIT they can be research, scientific, development, R&D, technical, and promotional. The keywords are searched in the entered data. Occurrences count as positive events (p) and everything else as negative events (n).

Fourth, p and n go to the information gain module [6], which computes the gain and shows it on screen. Fifth, the numerical gain is plotted so the analyst can see whether it has risen, fallen, or stayed constant.

Sixth, the graph is read. A positive slope means the data goes into the primary database. A zero or negative slope triggers a warning that the data has low value. That data is stored in the secondary database as history and is not used in result computation.

B. Building blocks

Fig. 1 shows the software architecture and Fig. 2 the flow chart of the stages. The building blocks are described below.

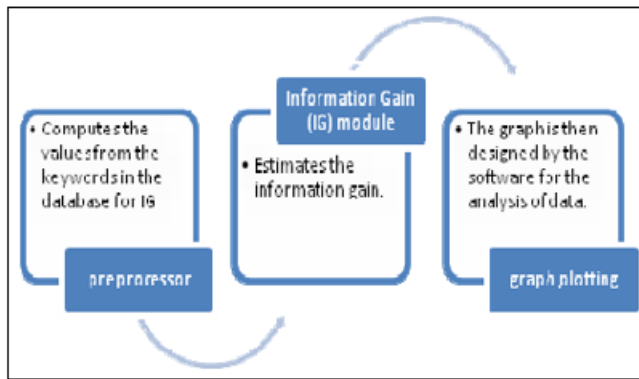


Figure 1: Proposed software architecture: preprocessor, information gain module, graph plotting

Preprocessor (Step 5.1): The keywords stored in the database are matched against the user's data. The result is evaluated as positive and negative events and passed to the information gain module.

Information gain module (Step 5.2): The formula from Shannon's information theory [6] gives the result. It is shown graphically and numerically so the system analyst can read it quickly.

Graph plotting (Step 5.3): The numerical gain is plotted and analyzed to separate the useful and relevant information from the mass of entered data.

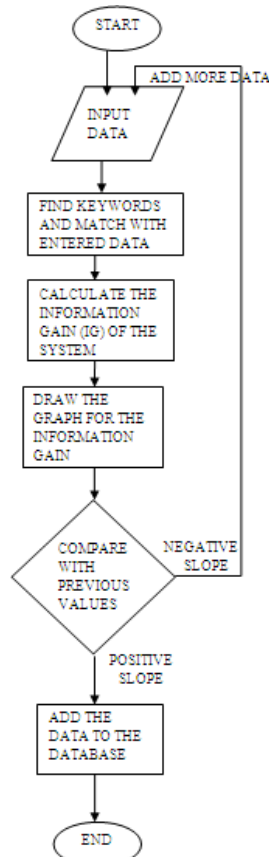


Figure 2: Flow chart for deploying information gain: input data; find keywords and match with entered data; calculate the IG; draw the graph; compare with previous values; positive slope adds the data to the database, negative slope asks for more data

C. Decision rule and the slope

The graph in Fig. 7 plots the gain against successive entries ("attempts"). The slope is the change from the previous values. Let $\Delta_t = IG_t - IG_{t-1}$ be the change at entry t . A positive trend over a window of recent entries means new entries still add information. A zero trend means saturation. A negative trend means new entries add less than earlier ones. The trend is a monitoring signal only. The per-record decision uses a threshold on the record score (Section 4.5(E)), tuned on validation data to meet a target recall.

4.4 Software Implementation

The modules were implemented on the Windows .NET platform in C#, because the module was meant to be integrated into the RFD software package under development at the DIT [34]. It would serve as the basis for reorganizing enterprise databases in a systematic way. The string values obtained from the user are used for database stuffing. This version handles text only. Computing gain for graphical data in JPEG form was left for later, and it would matter for several RFD applications. The software design, meaning the classes, their attributes, and their relationships, is shown in the class diagram in Fig. 3.

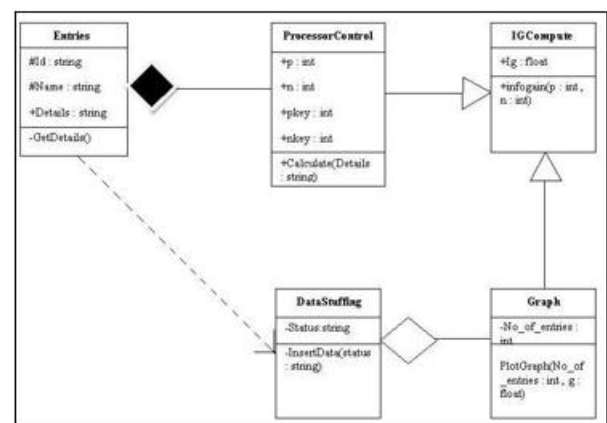


Figure 3: Class diagram

A. User interface

Version 1.0 opens with the introductory form in Fig. 4. The second form (Fig. 5) is the query form. The user enters a department, and the matching keywords are selected. The p and n values are then calculated from those predefined keywords.

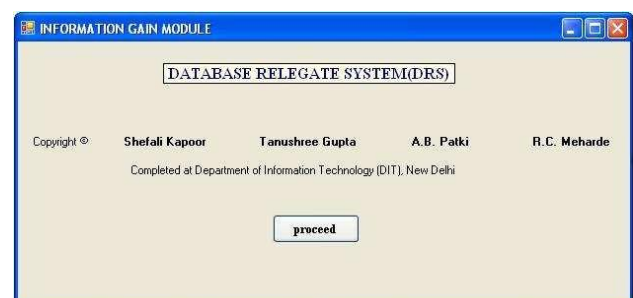


Figure 4: Introduction form.

Figure 5: Data collection form (preprocessor module)

B. Code modules

The pre-calculated p and n are passed to the function `infogain` (Code Module I), which returns the entropy of the textual data. The conditional entropy from Section 4.1 is computed with the same function by passing the p and n values of each keyword (Code Module II). The code is reproduced below.

```
double infogain (int p, int n)
{
    double info_content = 0;
    rem = ((double)p/(p+n)*Math.Log((double)p/(p+n),2) +
           ((double)n/(p+n)*Math.Log((double)n/(p+n),2)))
    return (rem);
}
```

Code Module I: entropy calculation.

```
Entropy = infogain(p, n);
Info_k1 = infogain(originalform.pkey1,
                  originalform.nkey1);
Info_k2 = infogain(originalform.pkey2,
                  originalform.nkey2);
Information_gain = (Entropy - (Info_k1 - Info_k2));
```

Code Module II: information gain computation.

For the sample problem of Section 4.1(B) the preprocessor finds $p = 5$ and $n = 12$, then $pkey1 = 1$, $nkey1 = 4$, $pkey2 = 3$, $nkey2 = 2$, $pkey3 = 1$, and $nkey3 = 4$. Passing these to the information gain module produces the gain shown in Fig. 6. The value is not rounded. All digits up to seven places are kept so that small changes in the IG computation stay visible.

Figure 6: IG calculation form (information gain module) (the displayed value is 0.1816255).

Fig. 7 plots the variation in the gain and gives the user a clear view of what the data contributes. The database is separated by IG value. A positive slope sends the data to the primary database for result computation. A negative slope, meaning a lower gain, does not delete the data. Those entries go to a secondary database (Fig. 8). All data is kept, and the partitioning avoids the overload.

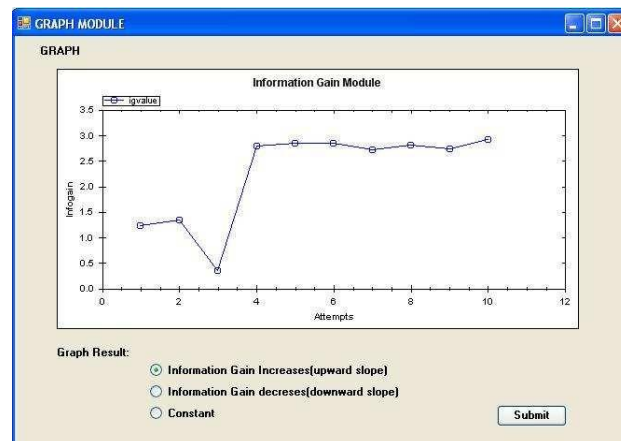


Figure 7: Graph plotting: IG value against successive attempts, with the options “Information Gain increases (upward slope)”, “decreases (downward slope)”, and “Constant”

```
private void radioButton1_CheckedChanged(object sender,
EventArgs e)
{
    string sqlINSERT = "insert into primary_db values
    (@uid,@uname,@details)";
    label = "Data added to primary database because
    information gain increases";
}
private void radioButton2_CheckedChanged(object sender,
EventArgs e)
{
    string sqlINSERT = "insert into secondary_db values
    (@uid,@uname,@details)";
    label = "Data added to secondary database because
    information gain decreases";
}
private void btnsubmit_Click(object sender, EventArgs e)
{
    string connstring = "Data Source=MPC-87016C7;Initial
    Catalog=project;Integrated Security=True";
    SqlConnection sqlconn = new SqlConnection(connstring);
    SqlCommand sqlcmd = new SqlCommand(sqlINSERT,
    sqlconn);
    sqlcmd.Parameters.Add("@uid", SqlDbType.Int).Value =
    originalform.id;
    sqlcmd.Parameters.Add("@uname",
    SqlDbType.VarChar).Value = originalform.name;
    sqlcmd.Parameters.Add("@details",
    SqlDbType.Text).Value = originalform.details;
    sqlconn.Open(); sqlcmd.ExecuteNonQuery();
    sqlconn.Close();
}
```

Code Module III: database stuffing.

Figure 8: Database stuffing.

C. Corrected implementation

The corrected functions compute the entropy with the negative-sum form and the conditional entropy as a weighted sum over a proper partition. The corrected insert uses one parameterized command with a configured connection string and disposable resources.

```

static double Entropy(int p, int n)
{
    if (p == 0 || n == 0) return 0.0;
    double t = p + n, a = p / t, b = n / t;
    return -(a * Math.Log(a, 2) + b * Math.Log(b, 2));
}

static double InformationGain(int p, int n, (int p, int
n[]) parts)
{
    double cond = 0.0;
    foreach (var s in parts)
        cond += (s.p + s.n) / (double)(p + n) *
        Entropy(s.p, s.n);
    return Entropy(p, n) - cond;    // always >= 0
}

```

Code Module IV (corrected). Entropy and information gain over a partition. Example: $\text{InformationGain}(6, 6, \text{new}[]\{(5,1),(1,5)\}) = 0.3500$; $\text{InformationGain}(5, 10, \text{new}[]\{(1,4),(3,2),(1,4)\}) = 0.1134$.

```

string table = isPrimary ? "primary_db" :
    "secondary_db";    // whitelist, never user input
using var conn = new SqlConnection(config.GetConnectionString
    tring("Retention"));
using var cmd = new SqlCommand($"INSERT INTO {table}
    (uid,      uname,      details)      VALUES
    (@uid,@uname,@details)",

```

```

conn);
cmd.Parameters.Add("@uid",      SqlDbType.Int).Value
    = record.Id;
cmd.Parameters.Add("@uname",    SqlDbType.VarChar).Value
    = record.Name;
cmd.Parameters.Add("@details",  SqlDbType.Text).Value
    = record.Details;
conn.Open(); cmd.ExecuteNonQuery();
Code Module V (corrected). Parameterized insert into the selected store.

```

4.5 Proposed AI-Enhanced Framework

Version 1.0 reduces information overload with an entropy computation over keywords. Advances in AI, machine learning, and natural language processing allow better performance, scale, and contextual understanding. This section keeps the architecture (preprocessor, IG module, partitioning; Fig. 1) and replaces the keyword matcher and the ad-hoc decision rule with the components in Fig. 9 and Table III.

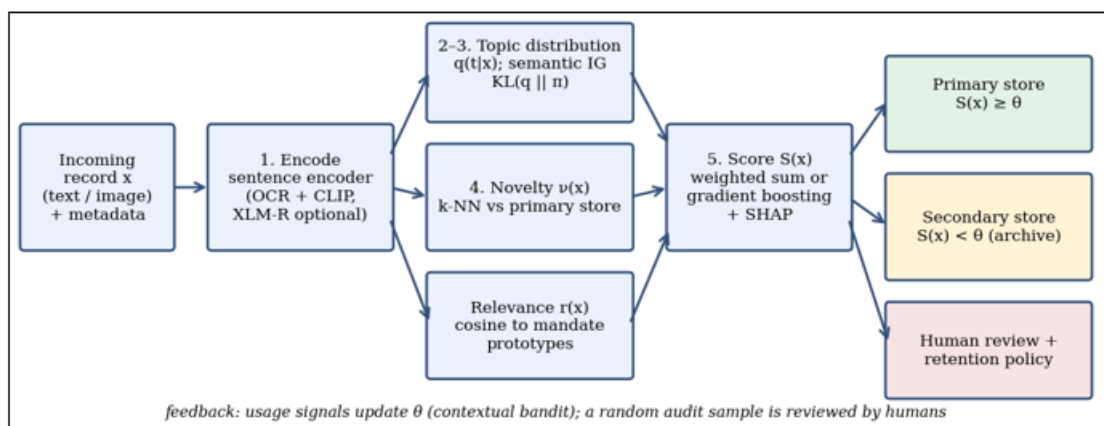


Figure 9: Processing pipeline of the proposed framework (new figure)

Table III: Pipeline stages of the proposed framework

Stage	Component	Output
1. Encode	Sentence encoder [20]; optional OCR + CLIP for images [22]; XLM-R for non-English text [21]	Embedding $e(x)$ per record
2. Topic distribution	Topic prototypes $c_1 \dots c_m$; softmax over cosine similarities	$q(\cdot x)$
3. Semantic IG	$KL(q(\cdot x) \parallel \pi)$, π the corpus topic distribution	$IG_sem(x)$, in bits
4. Novelty	k-NN search in the primary store (FAISS [25])	$v(x) = 1 - \max \cos(e(x), e(y))$
5. Score	Weighted sum or gradient boosting [35]; SHAP [28]	$S(x)$ and feature attributions
6. Route	Threshold θ (bandit-adapted); governance checks (Section 4.6)	Retain / archive / review

A. Semantic information gain

The main change is to define IG on meaning, not on exact keyword matches. Let $c_1 \dots c_m$ be topic prototypes. They can be embeddings of department keywords or cluster centroids of the existing corpus. Let $e(x)$ be the embedding of record x . The soft assignment to topics is $q(t|x) = \exp(\cos(e(x), c_t)/\tau) / \sum_u \exp(\cos(e(x), c_u)/\tau)$, where τ is a temperature. Let $\pi(t)$ be the average of $q(t|\cdot)$ over the current primary database. The **semantic information gain** of x is the KL divergence

$$IG_sem(x) = \sum_t q(t|x) \cdot \log_2 [q(t|x) / \pi(t)] \quad (\text{bits}).$$

$IG_sem(x)$ is never negative. It is large when x concentrates on topics the corpus holds little of. With $\pi = (0.4, 0.4, 0.2)$, a record with $q = (0.7, 0.2, 0.1)$ has $IG_sem = 0.265$ bits. A record with $q = (0.4, 0.38, 0.22)$ that mirrors the corpus has $IG_sem = 0.002$ bits. With hard assignments, where each keyword is present or absent, the quantity reduces to the counting form of Section 4.1, so version 1.0 is a special case. Two alternatives can be compared against it. One is the mutual information between a record's features and a retention label, estimated with k-NN estimators [36] or neural estimators [37]. The other is the entropy of the corpus topic distribution before and after adding x , which gives the marginal information used for monitoring in Section 4.3(C).

B. Context-aware text processing using transformer-based models

Version 1.0 detects relevant data with predefined keywords. A static list overlooks similar terms and paraphrased content. The enhanced model uses transformer encoders such as BERT (Bidirectional Encoder Representations from Transformers) [19], RoBERTa, or GPT-style models to analyze the whole input. Each input text becomes a contextual embedding, and cosine similarity against domain-specific vectors gives a relevance score.

Example 1. Input: "The cyber policy needs to be reformed." With sentence embeddings, phrases such as "Revising digital

security protocols” or “Upgrading cybersecurity frameworks” align with it even though they lack the words “cyber” and “policy.”

Example 2: Input: “Establishment of AI-based research labs in universities.” With no keyword expansion this may map to no predefined term. A transformer can link it to “R&D

initiatives” and “academic expansion,” which raises its IG value.

In the simulation of Section 5 the encoder is latent semantic analysis (LSA). It learns that surface forms co-occurring within a topic are related, but it is much weaker than a pretrained transformer. Table IV gives measured values from the simulation (seed 0) for four test records.

Table IV: Measured scores for four test records. $r(x)$: maximum cosine similarity to a mandate prototype; $v(x)$: novelty versus the primary store

Record type (simulation, seed 0)	$r(x)$	$v(x)$	IG sem (bits)	Keyword hits	Valuable (ground truth)
In-mandate, canonical wording	0.81	0.05	2.13	3	Yes
In-mandate, paraphrased wording	0.05	0.06	2.03	0	Yes
Off-mandate topic	0.12	0.16	6.21	1	No
Near-duplicate of an in-mandate record	0.83	0.01	2.04	3	No

Table IV shows the intended behaviour and one weak spot. The canonical in-mandate record has high relevance ($r = 0.81$). The near-duplicate has similar relevance but almost no novelty ($v = 0.01$), so a rule that needs both relevance and novelty archives it. The paraphrased in-mandate record has low relevance to the prototypes ($r = 0.05$) and no keyword hits, because the prototypes were built from canonical keywords. Section 5 measures this gap.

C. Semantic embedding and vector-based matching

Keyword matching gives way to semantic vector embeddings from models such as Sentence-BERT [20], GloVe, or fastText. Text is encoded into high-dimensional vectors and compared with cosine similarity. The IG module applies a similarity threshold, for example 0.80, to decide whether a semantic match counts toward information gain. Table V lists illustrative similarity values. They were not measured with a specific encoder.

Table V: Illustrative cosine similarity between input and keyword vectors (not measured)

Input sentence	Most similar keyword	Cosine similarity (illustrative)
“University expansion initiatives”	“Expansion”	0.89
“Government-funded research projects”	“R&D”	0.86
“Staff hiring and onboarding procedures”	“HR development”	0.72

D. Novelty and redundancy

The novelty term is $v(x) = 1 - \max \cos(e(x), e(y))$, where y ranges over the k nearest neighbours of x in the primary store. An approximate nearest-neighbour index computes it [25,26]. A value near 0 flags a near-duplicate. This is what the orthogonality discussion in Section 4.2 was after. MinHash [15] and SimHash [16] serve as lexical baselines.

E. ML-based relevance prediction and scoring

A supervised model such as Random Forest or XGBoost [35] can replace static thresholds and binary keyword matches. It is trained on labelled enterprise data to judge relevance. Features include semantic similarity scores, word and character n -gram frequencies, text length and structure, and document metadata such as timestamp and origin. Suppose the training data holds historical decisions on which records stayed in the primary database. The model learns that entries

mentioning government initiatives, innovation, or strategic outcomes are retained more often. A new entry on “5G deployment plans” can then be classed as highly relevant although that term was never predefined.

The record score combines the signals: $S(x) = f(\text{IG_sem}(x), v(x), r(x), m(x))$. Here $r(x)$ is the maximum similarity to a department prototype and $m(x)$ holds metadata. The simplest f is a weighted sum fitted on validation data. A supervised f uses gradient-boosted trees. A record is retained when $S(x) \geq \theta$. The threshold θ is chosen on a validation set to meet a target recall of valuable records, for example 95%, and is not a fixed constant.

Label bias and leakage. A model trained on historical retention decisions learns past habits, mistakes included. Anything never retained cannot be shown to have been valuable. Four measures limit the damage. Train on labels from an independent annotation exercise on a stratified sample. Hold out a random audit sample that skips the model and goes to human review. Report performance by category so that neglect of minority topics shows up. Keep the explanation with each decision.

A weighted sum cannot express “relevant and novel”, because raising v also raises the score of off-mandate records. A multiplicative rule can: $S_{\text{prod}}(x) = \sigma(w \cdot [\text{IG_sem}(x), r(x)] + b) \cdot v(x)$, where σ is the logistic function and the weights come from the $\text{IG_sem} + r$ model. A record scores high only if it is likely to be valuable and unlike the stored records. A simpler form, $r(x) \cdot v(x)$, is also tested. The lexical alternative replaces v with a MinHash test: a record whose estimated Jaccard similarity to any stored record is at least 0.5 is archived (Section 4.7).

F. Explainability via SHAP values

SHAP (SHapley Additive exPlanations) [28] attributes a decision to input features such as keyword presence and similarity scores, and can show the effect in summary plots or force plots. This supports trust in automated IG decisions. *Example (illustrative).* For a retained entry on “Blockchain adoption in government procurement,” a SHAP plot could show that the terms “adoption,” “government,” and “blockchain” carried most of the decision weight. Section 5.4 gives SHAP results measured in the simulation.

G. Adaptive data retention: reinforcement learning and contextual bandits

The first proposal used an RL agent, based on Q-learning or policy gradients [38], to tune retention policy. Its reward would come from downstream impact: was the data later queried in reports, and did it influence business outcomes? If entries mentioning “AI workforce development” are often cited in quarterly policy reviews, for example, the system learns to give similar incoming records higher IG scores.

Retention is a one-step decision, and its reward arrives late, so full RL is heavier than needed. A **contextual bandit** [27] fits better. The context is the record’s features and score. The actions are retain, archive, or send to human review. The reward combines the usage signal (was the record later retrieved or cited?) with a storage cost. Archived records are queried less just because they are archived, so the policy keeps an exploration rate ϵ : a random fraction is retained or reviewed regardless. Offline evaluation uses inverse-propensity weighting. The bandit is not part of the simulation in Section 5.

H. Real-time processing and big data architecture

Streaming ingestion needs a scalable, fault-tolerant stack. The proposal is Apache Kafka for ingestion, Apache Spark for distributed processing and ML pipelines [39], NoSQL stores such as MongoDB and Cassandra for unstructured data, and vector databases (FAISS [25], Pinecone [26]) for nearest-neighbour queries. Incoming project proposals from several government departments, for example, are scored for IG as they arrive, and each is routed to the primary or secondary layer. Section 5.5 reports the scoring latency measured in the simulation.

I. Integration of multimodal data (text and image)

Corporate data also holds scanned documents, charts, and images. CLIP [22,23] aligns text and image representations, and vision transformers [24] encode image content. With these the IG module can read an image from its caption or embedded text, check whether picture and text agree, and filter out irrelevant visuals. A scanned infographic titled “Cybersecurity Trends 2024” can be read with OCR plus CLIP. If its text contains “incident response,” “firewall architecture,” and “intrusion detection,” the CLIP embedding should align with cybersecurity vectors and raise its IG score. This extends the plan in Section 4.4 to move from text to JPEG data.

J. Integration with enterprise knowledge graphs

Knowledge graphs such as Neo4j [40] or RDF-based structures [41] connect entries through entities and relationships. Ontologies add meaning, and inference extracts more value from the data. An input mentioning “Machine Learning policy for education” can be linked to “AI programs,” “curriculum modernization,” and “national education initiatives,” which widens the context of the IG computation. Building and maintaining the ontology has a cost, so the graph is used as an enrichment feature for $r(x)$.

K. Multilingual data handling

International organizations log data in several languages. Multilingual transformers such as mBERT or XLM-R [21] let the IG module compute relevance across languages. A policy

written in Hindi or French can be matched to English keywords, so evaluation does not depend on the language.

L. Privacy-preserving IG computation

Sensitive applications call for federated learning [30] and differential privacy [29]. These keep IG computation within data regulations such as the GDPR and protect sensitive content while still computing aggregate relevance. In a federated IG calculation across departmental servers, for example, the central server receives only the computed gains, not the raw data.

M. Interactive dashboards and analyst tools

Analysts need tools to act on the scores. Dashboards, for example in Streamlit or Power BI [42], are proposed for real-time IG monitoring, drill-down by department, topic, or file, and what-if analysis on hypothetical text. An analyst could type a draft and see its projected IG score plotted in real time, with the contributing terms highlighted. Together these additions turn the IG system into a platform that scores data for value and connects to wider analytics, security, and multilingual needs.

N. Combined algorithm

Algorithm 1 gives the decision procedure for a stream of records.

```

Algorithm 1: retain/archive decision for record x
Input: x, primary store D, topic prototypes c, mandate
       prototypes M, model f, threshold  $\theta$ 
1  e ← Encode(x)
2  q ← softmax(cos(e, c_t)/ $\tau$ )
3  IG_sem ← KL(q ||  $\pi(D)$ )
4  v ← 1 - max over y in kNN(e, D) of cos(e, y)
5  r ← max over m in M of cos(e, m)
6  S ← f(IG_sem, v, r, metadata)
7  if S ≥  $\theta$ : add x to D and update  $\pi$  (retain)
8  else: store x in the secondary tier (archive)
9  log x, S,  $\theta$ , model version, SHAP attributions

```

Algorithm 1. Decision procedure of the proposed framework.

4.6 Retention Governance

Version 1.0 never deletes data. Low-gain records go to a secondary database. That principle stays, and it is made explicit here, because retention law (the IT Act, 2000 and the GDPR storage-limitation rules) means a model score is not a legal basis for deletion.

Records with $S(x) \geq \theta$ sit in the primary tier. They are used for result computation and indexed for search. Records with $S(x) < \theta$ go to the secondary tier. They are stored, excluded from result computation, and still searchable at lower priority, so a retrieval-augmented system [31] can find them when needed. Deletion happens only after the retention period has run out, with no legal hold in force and with approval from an authorized owner. Every deletion is logged.

Each routing decision stores the score, model version, threshold, and SHAP attributions. Humans review a random sample from time to time, and the findings feed back into evaluation.

Personal and sensitive data is classified before scoring, and the privacy options in Section 4.5(L) apply to every tier.

4.7 Evaluation Protocol and Simulation Setup

This section states the protocol and what was run. Two corpora are used. A synthetic corpus gives exact ground truth and controlled paraphrase and duplicate rates. A corpus of real package descriptions tests whether the findings survive text the simulator did not generate. Everything uses latent semantic analysis (LSA) as the encoder. No pretrained sentence encoder could be downloaded in the execution environment.

A. Synthetic corpus

A new corpus is generated for each of 30 random seeds. It has 12 topics with 40 concepts each. Concept popularity follows a Zipf distribution, and every concept has three surface forms, one canonical and two paraphrases. There are also 150 background words. Four topics are “in mandate” (the department’s scope) and carry 35% of the records.

Each seed has 6,000 records in time order, about 40 tokens each: 70% primary-topic words, 15% secondary-topic words, and 15% background. Thirty percent of records are *paraphrased*. They use canonical forms with probability 0.03 instead of 0.7, so keyword lists see them poorly. A binary source flag, correlated with mandate, serves as metadata. Another 20% of records, after the first 200, are noisy copies of an earlier record, with 5% of tokens deleted and 10% replaced by another surface form. A second scenario, run for 15 seeds, replaces 80% of tokens in the copies. Those duplicates keep their meaning but share far fewer exact tokens.

A record is *valuable* (retain) if it is on mandate **and** not a near-duplicate. Off-mandate and redundant records should be archived. Two simulated annotators flip each label independently with probability 0.08, which gives Cohen’s $\kappa = 0.66 \pm 0.01$. Models train on annotator A’s labels. Thresholds are set on a validation set with adjudicated, ground-truth labels. The first 1,500 records form the training set, the next 1,000 the validation set, and the last 3,500 the test stream. In training and validation the primary store is curated from the labels. In testing the store holds only the records the method routed to primary, so each decision feeds back into novelty and corpus statistics.

B. Real-text corpus

The real-text corpus was built from PyPI package descriptions retrieved through the PyPI JSON interface. Of 33,000 randomly sampled packages, 11,008 had at least one Topic classifier. After removing non-English text, descriptions with fewer than 40 content tokens, and packages with more than one topic label, 3,566 documents remained in the 12 most frequent topics. Software Development was split by its third-level label. Text was lower-cased and stripped of markup, code, links, and stop words, then truncated to 150 tokens.

For each of 20 seeds, 3,000 documents are sampled and a random mandate of three topics covering 20 to 40% of the documents is drawn. Then 750 noisy near-duplicates are inserted (5% of tokens deleted and 10% replaced by another word of the same topic), which gives 3,750 records. The department keywords are the six most distinctive frequent

terms of each mandate topic in the training portion, 25% of the stream. A record is valuable if its topic is in the mandate and it is not a near-duplicate. The mandate is a classifier-based proxy, so this corpus tests behaviour on real wording, not enterprise relevance. The source flag is random, so metadata carries no signal here.

C. Methods

The keyword IG baseline uses six canonical keywords per mandate topic. Each keyword is scored by its information gain against the training labels (Section 4.1(C)), and a record’s score is the sum over the keywords it contains. The TF-IDF prototype baseline takes the maximum cosine similarity to the mandate keyword prototypes. The BM25 baseline uses the BM25 score of the keyword query. The LDA baseline takes the KL divergence of a record’s 12-topic LDA distribution from the running primary-store distribution. Two duplicate filters serve as baselines for the novelty term. MinHash uses 128 permutations over token sets and archives a record when the estimated Jaccard similarity to any stored record reaches 0.5. SimHash uses 64-bit fingerprints and archives a record at a Hamming distance of 3 or less. Each filter is combined with the relevance score $r(x)$ (“MinHash dedup + r ”, “SimHash dedup + r ”) or with the IG_sem + r score (“IG_sem + r + MinHash dedup”).

The proposed method uses IG_sem with prototypes from k -means ($k = 12$) on the training embeddings and $\tau = 0.1$. The relevance $r(x)$ is the maximum cosine to the mandate keyword prototypes. The novelty $v(x)$ is measured against the primary store by 1-nearest neighbour. Logistic regression, a weighted sum, combines IG_sem and r (“IG_sem + r ”), or IG_sem, r , and v (“IG_sem + r + novelty”). Two multiplicative rules test the “relevant and novel” conjunction: $r(x) \cdot v(x)$, and $\sigma(\text{IG_sem} + r) \cdot v(x)$, where σ is the logistic score of the IG_sem + r model. The supervised scorer is a gradient-boosted tree model with 100 trees of depth 3 on IG_sem, v , and r , with and without the metadata features (length, source, distinct-token ratio).

Operating point. For every method, θ is the value that retains a target fraction of the valuable validation records. The target is 0.95 unless stated. For the main curated procedure, θ comes from a validation pass whose primary store is curated from the labels. The self-calibrated variant instead re-estimates θ on a second validation pass that follows the method’s own decisions, which matches the store the method sees in testing.

D. Metrics and statistics

The metrics are precision, recall, and F1 for the *retain* decision at θ , and average precision (AP, the area under the precision–recall curve) as a threshold-free summary. The archive rate is the share of test records sent to secondary storage. Retrieval quality is $n\text{DCG}@10$ over the final primary store, using 100 TF-IDF queries of three words from a mandate topic. The relevant documents are the valuable records of that topic. “Full store” means all records. Scoring latency per record is measured in a single process with NumPy and scikit-learn on a 4-core sandbox CPU. It excludes text encoding because the embeddings are precomputed.

Seeds are the unit of analysis. Each seed regenerates the corpus, annotators, and models, and every method sees the

same seed. Results are reported as mean $\pm$ standard deviation (SD). For selected pairs of methods, the per-seed differences in a metric are tested with a two-sided Wilcoxon signed-rank test. Confidence intervals of the mean difference use the t distribution with $n - 1$ degrees of freedom. The p -values are Holm-adjusted over the 28 comparisons reported for the synthetic corpus (and separately over the 28 for the real-text corpus).

E. Simulation parameters

Table VI lists the parameters needed to reproduce the simulation. The sentence-encoder path of the code and a loader for the 20 Newsgroups corpus are included in the supplementary files, but they were not run, because the dataset and model hosts were unreachable from the execution environment.

Table VI: Simulation parameters

Item	Setting
Seeds	Synthetic: 30 (0–29). Duplicates with 80% substitution: 15 (0–14). Temperature and dimension sensitivity: 10 (0–9). Real text: 20 (0–19)
Synthetic corpus	12 topics $\times$ 40 concepts $\times$ 3 surface forms; 150 background words; Zipf exponent 0.8; 6,000 records; length $\sim$ Poisson(40), minimum 15
Mandate (synthetic)	Topics 0–3 (35% of records); 6 canonical keywords per mandate topic drawn from the 15 most popular concepts
Paraphrase / duplicate / label noise	30% paraphrased (canonical-form probability 0.03 vs 0.7); 20% near-duplicates (5% deletions, 10% form substitutions; 80% in the second scenario); annotator flip rate 0.08
Splits (synthetic)	Train 1,500; validation 1,000; test 3,500 (time order)

Real corpus	PyPI descriptions; 3,000 sampled documents + 750 near-duplicates = 3,750 records; train 25%, validation 17%, rest test; mandate of 3 topics covering 20–40%; keywords: 6 per mandate topic
Embedding	TF-IDF (sublinear tf) $\rightarrow$ truncated SVD, 60 dimensions synthetic, 100 real (20 and 120 in sensitivity), L2-normalized
Topic prototypes	k-means, $k = 12$, on training embeddings; $\tau = 0.1$ (0.05, 0.2, 0.5 in sensitivity); π with additive smoothing 10^{-3}
Duplicate filters	MinHash: 128 permutations, Jaccard ≥ 0.5 ; SimHash: 64 bits, Hamming ≤ 3
Models	Weighted sum: logistic regression on standardized features. Supervised: gradient boosting, 100 trees, depth 3. LDA: 12 topics, 10 iterations
Threshold θ	Value retaining the target fraction (0.90, 0.95, 0.99) of valuable validation records; adjudicated validation labels
Retrieval	100 queries $\times$ 3 words from a mandate topic; TF-IDF cosine; relevance = valuable records of that topic; nDCG@10
Software	Python 3, NumPy, scikit-learn 1.7, SciPy, SHAP 0.49; 4-core CPU sandbox
Supplementary files	sim2.py (corpora and methods), run_*.py, summarize2.py, build_corpus.py, make_20ng_corpus.py, pypi_corpus.json, per-seed result files

5. Results & Discussion

5.1 Main comparison (synthetic corpus)

Table VII: Results at a 95% recall target on the synthetic corpus (mean $\pm$ SD over 30 seeds). The nDCG@10 of the full store (all 6,000 records) is 0.808 ± 0.018 . Static baselines retain all test records, so their nDCG@10 reflects the curated training and validation records plus the full test stream. Latency excludes encoding; static baselines are scored in batch

Method	Precision	Recall	F1	AP	nDCG@10	Latency (ms)
Keyword IG (corrected)	0.281 ± 0.008	1.000 ± 0.000	0.439 ± 0.009	0.654 ± 0.019	0.864 ± 0.015	n/a
TF-IDF prototype	0.281 ± 0.008	1.000 ± 0.000	0.439 ± 0.009	0.678 ± 0.018	0.864 ± 0.015	n/a
BM25 prototype	0.281 ± 0.008	1.000 ± 0.000	0.439 ± 0.009	0.679 ± 0.018	0.864 ± 0.015	n/a
LDA topic IG	0.666 ± 0.110	0.913 ± 0.045	0.766 ± 0.088	0.683 ± 0.121	0.879 ± 0.015	0.201
MinHash dedup + r	0.416 ± 0.038	0.950 ± 0.015	0.578 ± 0.036	0.885 ± 0.014	0.988 ± 0.004	0.253
SimHash dedup + r	0.366 ± 0.033	0.950 ± 0.015	0.527 ± 0.034	0.783 ± 0.019	0.881 ± 0.013	0.271
IG sem + r (proposed)	0.778 ± 0.054	0.964 ± 0.029	0.860 ± 0.040	0.833 ± 0.021	0.865 ± 0.016	0.195
IG sem + r + novelty	0.780 ± 0.055	0.961 ± 0.029	0.860 ± 0.040	0.866 ± 0.035	0.868 ± 0.016	0.193
IG sem + r + MinHash	0.963 ± 0.070	0.963 ± 0.029	0.962 ± 0.044	0.987 ± 0.018	0.990 ± 0.004	0.178
$\sigma(\text{IG sem} + r) \times v$	0.830 ± 0.165	0.917 ± 0.022	0.863 ± 0.104	0.916 ± 0.063	0.970 ± 0.009	0.19
Supervised S(x) (no metadata)	0.909 ± 0.122	0.906 ± 0.024	0.902 ± 0.087	0.941 ± 0.054	0.975 ± 0.010	0.322

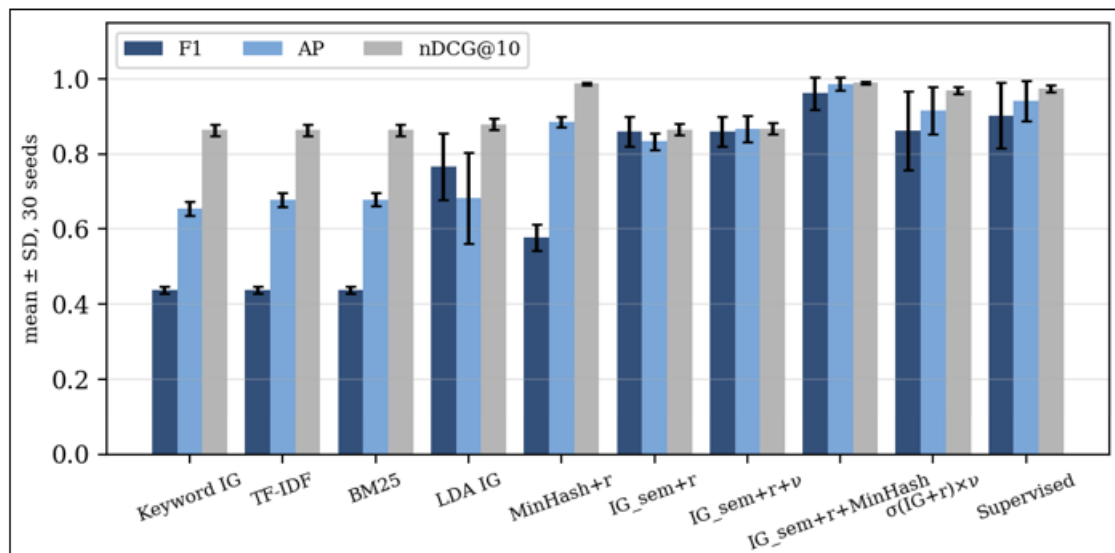


Figure 10: F1, AP, and nDCG@10 by method at the 95% recall target (mean $\pm$ SD over 30 seeds).

Lexical baselines cannot archive. For keyword IG, TF-IDF, and BM25, the threshold that retains 95% of valuable validation records is zero. They keep every record, so the archive rate is 0% and precision (0.281) equals the base rate of valuable records. Paraphrasing is the cause. 30% of valuable records are paraphrased, and 78% of those contain no department keyword.

Semantic IG with a relevance term. IG_sem + r reaches F1 = 0.860 ± 0.040 and AP = 0.833 ± 0.021 and archives 65% of test records. Its F1 is higher than the keyword baseline by +0.421 (95% CI 0.406 to 0.436, Holm $p < 0.001$), and it also beats LDA topic IG, which has no notion of mandate, by +0.093 (95% CI 0.057 to 0.129, Holm $p < 0.001$). Achieved recall (0.964) is close to the 0.95 target.

Duplicate filters. MinHash dedup + r archives 96% of redundant records, but its F1 (0.578) is low because r alone

ranks relevance poorly. Combining the MinHash filter with the IG_sem + r score gives the best result in this simulation: F1 0.962, AP 0.987, 97% of redundant records archived, and nDCG@10 0.990. The embedding-based rule $\sigma(\text{IG_sem} + \text{r}) \times \text{v}$ reaches F1 0.863 and archives 84% of redundant records. Its F1 is lower than the MinHash combination by -0.099 (95% CI -0.128 to -0.070 , Holm $p < 0.001$). SimHash with a Hamming threshold of 3 archives only 19% of redundant records, because 10% of the tokens in each copy differ.

Supervised scorer. The gradient-boosted scorer reaches F1 0.902 ± 0.087 and AP 0.941 ± 0.054 . Its F1 exceeds that of IG_sem + r by +0.043 (95% CI 0.017 to 0.069, Holm $p < 0.001$), but its seed-to-seed SD is high, and in 87% of seeds its achieved recall fell below 0.93, against 17% for IG_sem + r.

5.2 Ablations

Table VIII: Ablations at the 95% recall target on the synthetic corpus (mean $\pm$ SD over 30 seeds). “Redundant archived”: share of on-mandate near-duplicates routed to secondary storage

Variant	F1	Recall	AP	Archive rate	Redundant archived
IG sem only	0.85 ± 0.05	0.95 ± 0.07	0.74 ± 0.09	65%	5%
Novelty only	0.46 ± 0.01	0.97 ± 0.01	0.26 ± 0.02	9%	2%
Relevance r only	0.52 ± 0.03	0.95 ± 0.01	0.76 ± 0.02	25%	4%
IG sem + novelty (no r)	0.85 ± 0.06	0.94 ± 0.09	0.80 ± 0.12	66%	7%
IG sem + r (proposed)	0.86 ± 0.04	0.96 ± 0.03	0.83 ± 0.02	65%	3%
IG sem + r + novelty	0.86 ± 0.04	0.96 ± 0.03	0.87 ± 0.03	65%	4%
$\text{r} \times \text{v}$	0.52 ± 0.02	0.94 ± 0.02	0.83 ± 0.02	26%	23%
$\sigma(\text{IG_sem} + \text{r}) \times \text{v}$	0.86 ± 0.10	0.92 ± 0.02	0.92 ± 0.06	67%	84%
Supervised S(x) (no metadata)	0.90 ± 0.09	0.91 ± 0.02	0.94 ± 0.05	71%	87%
Supervised S(x) (with metadata)	0.90 ± 0.09	0.91 ± 0.02	0.95 ± 0.05	71%	87%

IG_sem and relevance work together. Relevance alone is far weaker (F1 0.520). For IG_sem alone, the F1 difference to IG_sem + r is not significant (+0.005 (95% CI -0.006 to 0.016 , Holm $p > 1.000$)), but AP is higher with r (+0.089 (95% CI 0.060 to 0.119 , Holm $p < 0.001$)).

Novelty adds little in a weighted sum. Adding v to IG_sem + r changes AP by +0.033 (95% CI 0.025 to 0.041, Holm $p < 0.001$) and the share of redundant records archived by +0.016 (95% CI 0.009 to 0.023, Holm $p < 0.001$), which is statistically reliable but small in size, and F1 does not change.

A linear score cannot express “relevant but not novel”, because raising v also raises the score of off-mandate records. A multiplicative rule can. $\sigma(\text{IG_sem} + \text{r}) \times \text{v}$ archives 84% of redundant records, +0.794 (95% CI 0.771 to 0.818 , Holm $p < 0.001$) more than the weighted sum, and raises nDCG@10 by +0.102 (95% CI 0.096 to 0.108 , Holm $p < 0.001$), with the same F1 (+0.003 (95% CI -0.027 to 0.032 , Holm $p > 1.000$)). The simpler product $\text{r} \times \text{v}$ fails (F1 0.518), because r alone ranks relevance too poorly. Adding the metadata features to the supervised scorer changes F1 by -0.001 (95% CI -0.008 to 0.005 , Holm $p > 1.000$).

5.3 Duplicate handling when copies are paraphrased

Table IX: Effect of the edit rate in duplicate records on F1 and on the share of redundant records archived (95% recall target).
Left: 30 seeds. Right: 15 seeds with 80% of tokens replaced in the copies

Method	F1 (copies 10% edited)	Redundant archived	F1 (copies 80% edited)	Redundant archived
IG sem + r (proposed)	0.86	3%	0.86	0%
IG sem + r + novelty	0.86	4%	0.86	0%
$\sigma(\text{IG sem} + r) \times v$	0.863	84%	0.793	3%
Supervised S(x) (no metadata)	0.902	87%	0.865	58%
IG sem + r + MinHash	0.962	97%	0.881	22%
MinHash dedup + r	0.578	96%	0.526	22%
SimHash dedup + r	0.527	19%	0.515	0%

When 80% of the tokens in a copy are replaced, the MinHash combination archives only 22% of redundant records, down from 97%. This is the case in which semantic novelty should help, because the copies keep their meaning. It did not help with LSA embeddings: the product rule archives 3% and its F1 falls to 0.793. Only the supervised scorer archives a meaningful share (58%). So the claim that embedding novelty

beats a lexical filter on paraphrased duplicates is not supported with this encoder. Whether a pretrained sentence encoder changes that is untested.

5.4 Sensitivity to the target recall, temperature, and encoder size

Table X: Effect of the target recall on the operating point on the synthetic corpus (mean over 30 seeds)

Method	Target recall	Precision	Recall	F1	Archive rate
Keyword IG (corrected)	0.9	0.281	1	0.439	0%
	0.95	0.281	1	0.439	0%
	0.99	0.281	1	0.439	0%
IG sem + r (proposed)	0.9	0.776	0.932	0.846	66%
	0.95	0.778	0.964	0.86	65%
	0.99	0.777	0.987	0.868	64%
Supervised S(x) (no metadata)	0.9	0.928	0.855	0.887	74%
	0.95	0.909	0.906	0.902	71%
	0.99	0.748	0.968	0.823	58%

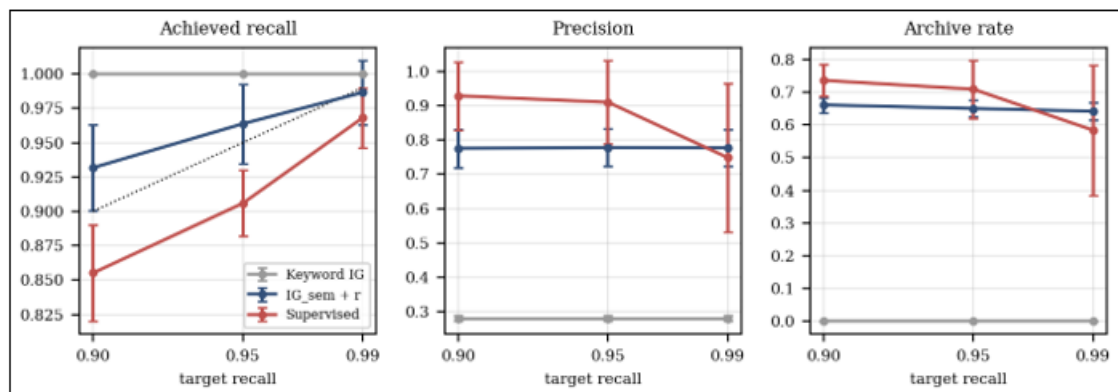


Figure 11: Achieved recall, precision, and archive rate as the target recall varies (mean $\pm$ SD over 30 seeds; dotted line: achieved = target).

Table XI: Sensitivity of F1 to the softmax temperature and the embedding dimension (mean over 10 seeds; 95% recall target)

Setting (F1)	IG sem + r	$\sigma(\text{IG sem} + r) \times v$	Supervised (no metadata)
$\tau = 0.05$	0.854	0.805	0.879
$\tau = 0.1$ (default)	0.854	0.856	0.872
$\tau = 0.2$	0.816	0.801	0.824
$\tau = 0.5$	0.794	0.695	0.779
LSA dim = 20	0.857	0.593	0.873
LSA dim = 60 (default)	0.854	0.856	0.872
LSA dim = 120	0.849	0.921	0.882

For IG sem + r, precision stays at 0.78 to 0.78 as the target recall rises from 0.90 to 0.99, and achieved recall (0.93, 0.96, 0.99) follows the target. The archive rate drops only from 66% to 64%. The supervised scorer loses F1 at the 99% target (0.823). Higher temperatures flatten the topic distributions

and lower F1 for all three methods. The embedding dimension barely affects IG sem + r, but the product rule depends strongly on it (F1 0.593 at 20 dimensions and 0.921 at 120), because its novelty term relies on the quality of nearest-neighbour distances.

5.5 Threshold calibration

Table XII: Effect of calibrating θ on a validation pass that follows the method's own decisions (synthetic corpus, 30 seeds, 95% recall target)

Method	F1 (curated θ)	F1 (self-calibrated θ)	Recall (curated θ)	Recall (self-calibrated θ)
IG sem + r (proposed)	0.86 ± 0.04	0.86 ± 0.04	0.96 ± 0.03	0.96 ± 0.03
$\sigma(\text{IG sem} + r) \times v$	0.86 ± 0.10	0.82 ± 0.12	0.92 ± 0.02	0.91 ± 0.02
Supervised S(x) (no metadata)	0.90 ± 0.09	0.90 ± 0.10	0.91 ± 0.02	0.91 ± 0.03

The supervised scorer misses its recall target and varies between seeds. The suspected cause was a mismatch between the curated store used to set θ and the self-built store used in testing. Self-calibration does not fix it: F1 changes by +0.001 (95% CI -0.005 to 0.006, Holm p 1.000) and recall by +0.000 (95% CI -0.006 to 0.006, Holm p 1.000). For the product rule it lowers F1 by -0.039 (95% CI -0.052 to -0.027, Holm p <

0.001). The store mismatch is therefore not the main cause of the instability. The cause was not identified. Label noise (8%) and the small training set for the tree model are candidates, and neither was tested.

5.6 Statistical tests

Table XIII: Paired comparisons over 30 seeds on the synthetic corpus (Wilcoxon signed-rank test, Holm-adjusted over 28 comparisons; t-based confidence intervals)

Method A	Method B	Metric	A - B	95% CI	Holm p
IG sem + r	Keyword IG	F1	0.421	0.406 to 0.436	< 0.001
IG sem + r	Keyword IG	AP	0.18	0.172 to 0.188	< 0.001
IG sem + r	IG sem only	AP	0.089	0.060 to 0.119	< 0.001
IG sem + r + novelty	IG sem + r	AP	0.033	0.025 to 0.041	< 0.001
IG sem + r + novelty	IG sem + r	Redundant archived	0.016	0.009 to 0.023	< 0.001
$\sigma(\text{IG sem} + r) \times v$	IG sem + r + novelty	Redundant archived	0.794	0.771 to 0.818	< 0.001
$\sigma(\text{IG sem} + r) \times v$	IG sem + r + novelty	nDCG@10	0.102	0.096 to 0.108	< 0.001
$\sigma(\text{IG sem} + r) \times v$	IG sem + r + novelty	F1	0.003	-0.027 to 0.032	1.000
IG sem + r + MinHash	IG sem + r	Redundant archived	0.938	0.929 to 0.948	< 0.001
$\sigma(\text{IG sem} + r) \times v$	IG sem + r + MinHash	F1	-0.099	-0.128 to -0.070	< 0.001
Supervised (no meta.)	IG sem + r	F1	0.043	0.017 to 0.069	< 0.001
Supervised (no meta.)	$\sigma(\text{IG sem} + r) \times v$	F1	0.04	0.003 to 0.077	1
Supervised (meta.)	Supervised (no meta.)	F1	-0.001	-0.008 to 0.005	1.000
Supervised, self-calibrated	Supervised (no meta.)	F1	0.001	-0.005 to 0.006	1.000

Large differences are reliable: semantic IG with relevance against keyword scoring, the multiplicative rule against the weighted sum for redundancy, and MinHash filtering against the embedding rules. Several differences are not: IG sem against IG sem + r in F1, novelty in the weighted sum in F1, the product rule against the weighted sum in F1, and metadata

in the supervised scorer. The seeds share one generator, so the tests describe variation of the simulator, not variation of enterprise data.

5.7 Results by category and SHAP stability

Table XIV: Recall of valuable test records by wording style at the 95% recall target (synthetic corpus, mean over 30 seeds). The keyword baseline's recall of 1.00 comes from retaining every record (Table VII)

Method (recall of valuable records)	Canonical wording	Paraphrased wording	Redundant archived
Keyword IG (corrected)	1	1	0%
Relevance r only	1	0.83	4%
IG sem + r (proposed)	1	0.88	3%
IG sem + r + novelty	1	0.87	4%
$\sigma(\text{IG sem} + r) \times v$	1	0.73	84%
Supervised S(x) (no metadata)	0.95	0.81	87%
IG sem + r + MinHash	1	0.88	97%

Paraphrased wording is the weakest category for the semantic methods. IG sem + r recalls 0.88 of paraphrased valuable records against 1.00 for canonical wording, which fits prototypes built from canonical keywords. For the supervised scorer with metadata, the mean absolute SHAP values on the validation set were IG sem 1.49, novelty v 0.53, relevance r

0.48, length 0.11, source 0.04, distinct-token ratio 0.14. IG sem dominates, and the metadata features contribute little. The feature ranking is stable across seeds (mean pairwise Spearman correlation 0.91).

5.8 Real-text corpus

Table XV: Results on the real-text corpus of PyPI descriptions at a 95% recall target (mean over 20 seeds; base rate of valuable records 0.248 ± 0.013 ; nDCG@10 of the full store 0.403 ± 0.075). SDs are in the supplementary results

Method	Precision	Recall	F1	AP	nDCG@10	Redundant archived
Keyword IG (corrected)	0.248	1	0.397	0.358	0.484	0%
TF-IDF prototype	0.248	1	0.397	0.366	0.484	0%
BM25 prototype	0.248	1	0.397	0.363	0.484	0%
LDA topic IG	0.262	0.949	0.41	0.25	0.515	30%
MinHash dedup + r	0.284	0.943	0.437	0.467	0.535	95%
IG sem + r (proposed)	0.268	0.947	0.418	0.446	0.491	7%
IG sem + r + novelty	0.274	0.932	0.423	0.454	0.495	16%
IG sem + r + MinHash	0.283	0.934	0.434	0.465	0.536	95%
$\sigma(\text{IG sem} + r) \times v$	0.27	0.926	0.417	0.37	0.517	67%
Supervised S(x) (no metadata)	0.267	0.949	0.416	0.397	0.503	31%
Supervised S(x) (with metadata)	0.277	0.95	0.429	0.43	0.511	32%

The advantage of semantic scoring is much smaller on real text. The base rate of valuable records is 0.248, and F1 for keyword IG is 0.397 because every record is retained. 57% of valuable records contain no department keyword. IG sem + r reaches F1 0.418 and AP 0.446. Against keyword IG, the F1 difference is +0.020 (95% CI 0.015 to 0.026, Holm $p < 0.001$) and the AP difference is +0.089 (95% CI 0.080 to 0.097, Holm $p < 0.001$). These are reliable but small. LSA embeddings fitted on about 900 short documents rank relevance poorly: AP stays below 0.5 for every method.

Duplicates behave as they did in the synthetic corpus. MinHash dedup + r reaches the highest F1 (0.437) and archives 95% of redundant records. The IG sem + r + MinHash combination archives 95%. The embedding novelty term in the weighted sum changes F1 by +0.005 (95% CI 0.003 to 0.007, Holm $p < 0.001$) and archives 16%. The product rule archives 67%, which is lower than MinHash by -0.275 (95% CI -0.440 to -0.111, Holm $p = 0.274$). The supervised scorer is no better than IG sem + r on this corpus (-0.002 (95% CI -0.008 to 0.005, Holm $p = 1.000$)). The metadata flag is random here, and the supervised scorer with metadata gains +0.013 (95% CI 0.008 to 0.017, Holm $p < 0.001$) in F1, probably through text length.

5.9 Cost

Scoring one record takes about 0.195 ms with the weighted-sum score, 0.178 ms when the MinHash filter is added, and 0.322 ms with the gradient-boosted scorer, on one CPU process. That includes the k-NN and MinHash searches over a store of up to roughly two thousand records. The figures exclude encoding. They will not carry over to production-size stores or pretrained transformer encoders, where inference dominates the cost, and stores of millions of records need an approximate nearest-neighbour index. At the 95% recall target, IG sem + r archives about 65% of the stream, the MinHash combination 72%, and the supervised scorer 71%.

5.10 Comparison with previous work

The direct predecessor is the keyword-based IG module of version 1.0. Its corrected form is the keyword IG baseline here, and it kept every record at a 95% recall target on the synthetic corpus (F1 0.439). Semantic IG with a relevance term reached F1 0.860, a gain of 42.1 F1 points. On real text the gain was 2.0 points. TF-IDF [11] and BM25 [12] behaved like keyword IG on both corpora. That agrees with the general

finding that lexical matching fails on paraphrased text, which is why sentence embeddings were introduced [20].

MinHash [15] and SimHash [16] are the established answers to redundancy, and they were run here. For near-exact copies MinHash was better than the embedding novelty term on both corpora. For heavily edited copies neither worked. Feature-selection studies report IG as a competitive term-scoring criterion [9]. The simulation does not test that claim, because IG is applied to records and not to terms.

5.11 Limitations and threats to validity

The synthetic corpus comes from a generator built around the ideas the method uses: mandate, paraphrase, and duplicates. The results show how the methods behave when those effects exist. They do not show how common or severe the effects are in enterprise data, and the ranking of methods will shift with the paraphrase rate (30%), the duplicate rate (20%), and the label noise (8%). The real-text corpus is real wording, but its mandate is a package-classifier proxy and its duplicates are injected. No enterprise data was used.

Every result uses LSA embeddings. Pretrained sentence encoders were not available, so whether they narrow the paraphrase gap, or make the embedding novelty term competitive with MinHash, is untested. Thresholds and τ depend on the encoder. The contextual-bandit policy, concept drift, the multilingual and OCR/CLIP extensions, Recall@k, and production-scale throughput were not simulated. Validation labels are adjudicated while training labels are noisy, because a 95% recall target means little when many “positives” are mislabelled negatives. The test store depends on each method’s own decisions, so methods do not see identical data. The instability of the supervised scorer is unexplained (Section 5.5). Finally, IG and novelty measure statistical surprise, not business importance. A rare but critical record, such as a compliance notice, may score low, so human review and the governance tiers of Section 4.6 stay necessary.

6. Conclusion

The first version of this work gave a schematic plan to cut information overload in enterprise databases that hold a surplus of irrelevant data. It looked for the information in user-entered data that raises the overall information gain of the system. It proposed a probabilistic approach that partitions the database instead of deleting data.

This paper corrects the mathematics and reference code of that entropy-based module. It redefines information gain on embedding-derived topic distributions, adds relevance, novelty, and supervised scoring, and specifies a governance model and an evaluation protocol. The protocol was run on 30 seeds of a synthetic corpus, 15 seeds of a harder duplicate scenario, and 20 seeds of a corpus of real package descriptions. Four findings follow.

Lexical scoring cannot cut volume at a high recall target. Keyword IG, TF-IDF, and BM25 had to keep every synthetic record to retain 95% of the valuable ones, so their precision (0.281) equals the base rate. Paraphrased valuable records contain few department keywords.

Semantic IG with a relevance term is cheap and helps, but how much depends on the text. On the synthetic corpus it reached F1 0.860 against 0.439 and archived about 65% of the stream. On real package descriptions it reached F1 0.418 against 0.397. LSA embeddings rank relevance poorly on short real documents, and paraphrased records remain its weak point (recall 0.88 on the synthetic corpus).

Redundancy is handled best by a lexical filter when copies are near-exact. IG_{sem} + r with a MinHash filter reached F1 0.962 and archived 97% of redundant synthetic records. On real text MinHash dedup + r had the highest F1 (0.437). The embedding novelty term added almost nothing in a weighted sum, and the multiplicative rule archived 84%, less than MinHash. When 80% of the tokens in a copy were replaced, no unsupervised rule worked, and only the supervised scorer archived a meaningful share (58%).

The supervised scorer has the highest AP on the synthetic corpus (0.941) and raises retrieval nDCG@10 to 0.975 (full store: 0.808). It is unstable between seeds and misses the recall target in most of them, and calibrating θ on its own decisions did not change that.

The earlier draft of the AI-enhanced module claimed four capabilities. Semantic awareness is supported on the synthetic corpus and only weakly on real text with a weak encoder. Scalability is a design only: scoring latency was measured, but no pipeline was benchmarked. Adaptivity is designed as a contextual bandit and was not simulated. Transparency through SHAP is supported, since the feature ranking was stable across seeds (Spearman 0.91).

7. Future Scope

The main limits are the LSA encoder, the absence of enterprise data, and the parts of the framework that were designed but not simulated. The first step is to repeat the protocol with a pretrained sentence encoder. The supplementary files include a loader for 20 Newsgroups and an encoder switch for this purpose, which were not run here. That run would show whether embedding novelty can match MinHash on paraphrased copies. Other steps are testing the contextual bandit, concept drift, the multilingual and OCR extensions, and the privacy leakage of shared scores, and finding the cause of the supervised scorer's instability.

Against the historical keyword method, the proposed approach scores meaning and novelty instead of word counts, explains each decision, aims at a stated recall, and never deletes on a score alone. In these experiments the cheapest combination, IG_{sem} + r with a MinHash filter, performed at or near the best on both corpora. The simulation code, the real-text corpus, and the raw per-seed results are provided as supplementary material.

Acknowledgement

The authors would like to show gratitude towards the engineering college authorities for allowing the student to work at the DIT, and owe their deepest gratitude to the experienced and helpful people at the DIT. The earlier work undertaken at the DIT in evidence-based software engineering was useful for a quick start towards this project.

References

- [1] Chowdhary S, Joshi S, Patki T, Sharma V, Chowdhary A, Patki AB, et al. Cyber crime and insurance industries. In: Proceedings of the 4th National Conference on Management using Evidence Based Software Engineering. New Delhi: INDIACOM; 2010 Feb 25-26.
- [2] Quinlan JR. Induction of decision trees. *Mach Learn*. 1986;1(1):81-106.
- [3] Moore A. Information gain [Internet]. Pittsburgh: Carnegie Mellon University, School of Computer Science. Available from: <https://www.cs.cmu.edu/afs/cs/Web/People/awm/tutorials/infogain.html>
- [4] Gupta T, Kapoor S. Software development for IG. Summer internship report no. DIT/DIR(RCM)/01. Department of Information Technology; 2010 Jun.
- [5] Kapoor S, Gupta T, Patki AB, Meharde RC. Applications of information gain methodology for corporate databases.
- [6] Shannon CE, Weaver W. The mathematical theory of communication. Urbana: University of Illinois Press; 1949.
- [7] Cover TM, Thomas JA. Elements of information theory. 2nd ed. Hoboken (NJ): Wiley; 2006.
- [8] Russell S, Norvig P. Artificial intelligence: a modern approach. Upper Saddle River (NJ): Prentice Hall.
- [9] Yang Y, Pedersen JO. A comparative study on feature selection in text categorization. In: Proceedings of the 14th International Conference on Machine Learning. 1997. p. 412-20.
- [10] Andrews J, Benisch M, Sardinha A, Sadeh N. Using information gain to analyze and fine tune the performance of supply chain trading agents. In: Agent-mediated electronic commerce and trading agent design and analysis. Berlin: Springer; 2009.
- [11] Salton G, Buckley C. Term-weighting approaches in automatic text retrieval. *Inf Process Manag*. 1988;24(5):513-23.
- [12] Robertson S, Zaragoza H. The probabilistic relevance framework: BM25 and beyond. *Found Trends Inf Retr*. 2009;3(4):333-89.
- [13] Blei DM, Ng AY, Jordan MI. Latent Dirichlet allocation. *J Mach Learn Res*. 2003;3:993-1022.

- [14] Grootendorst M. BERTopic: neural topic modeling with a class-based TF-IDF procedure. arXiv:2203.05794 [Preprint]. 2022.
- [15] Broder AZ. On the resemblance and containment of documents. In: Proceedings of Compression and Complexity of Sequences. IEEE; 1997. p. 21-9.
- [16] Charikar MS. Similarity estimation techniques from rounding algorithms. In: Proceedings of the 34th ACM Symposium on Theory of Computing. 2002. p. 380-8.
- [17] Itti L, Baldi P. Bayesian surprise attracts human attention. Vision Res. 2009;49(10):1295-306.
- [18] Moreno PJ, Ho PP, Vasconcelos N. A Kullback-Leibler divergence based kernel for SVM classification in multimedia applications. In: Advances in Neural Information Processing Systems 16. 2004. p. 1385-92.
- [19] Devlin J, Chang MW, Lee K, Toutanova K. BERT: pre-training of deep bidirectional transformers for language understanding. arXiv:1810.04805 [Preprint]. 2018.
- [20] Reimers N, Gurevych I. Sentence-BERT: sentence embeddings using Siamese BERT-networks. arXiv:1908.10084 [Preprint]. 2019.
- [21] Conneau A, Khandelwal K, Goyal N, Chaudhary V, Wenzek G, Guzmán F, et al. Unsupervised cross-lingual representation learning at scale. arXiv:1911.02116 [Preprint]. 2019.
- [22] Radford A, Kim JW, Hallacy C, Ramesh A, Goh G, Agarwal S, et al. Learning transferable visual models from natural language supervision. In: Proceedings of the 38th International Conference on Machine Learning. 2021.
- [23] OpenAI. CLIP: connecting text and images [Internet]. 2021. Available from: <https://openai.com/research/clip>
- [24] Dosovitskiy A, Beyer L, Kolesnikov A, Weissenborn D, Zhai X, Unterthiner T, et al. An image is worth 16x16 words: transformers for image recognition at scale. arXiv:2010.11929 [Preprint]. 2020.
- [25] Johnson J, Douze M, Jégou H. Billion-scale similarity search with GPUs. IEEE Trans Big Data. 2021;7(3):535-47.
- [26] Pinecone Systems Inc. Vector search for AI applications [Internet]. 2023. Available from: <https://www.pinecone.io>
- [27] Li L, Chu W, Langford J, Schapire RE. A contextual-bandit approach to personalized news article recommendation. In: Proceedings of the 19th International Conference on World Wide Web. 2010. p. 661-70.
- [28] Lundberg SM, Lee SI. A unified approach to interpreting model predictions. In: Advances in Neural Information Processing Systems 30. 2017. p. 4765-74.
- [29] Abadi M, Chu A, Goodfellow I, McMahan HB, Mironov I, Talwar K, et al. Deep learning with differential privacy. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. 2016. p. 308-18.
- [30] McMahan B, Moore E, Ramage D, Hampson S, Arcas BA. Communication-efficient learning of deep networks from decentralized data. In: Proceedings of the 20th International Conference on Artificial Intelligence and Statistics. 2017. p. 1273-82.
- [31] Lewis P, Perez E, Piktus A, Petroni F, Karpukhin V, Goyal N, et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. In: Advances in Neural Information Processing Systems 33. 2020. p. 9459-74.
- [32] Walsh JL. A closed set of normal orthogonal functions. Am J Math. 1923;45(1):5-24.
- [33] Department of Information Technology, Government of India. Results Framework Document (RFD) [Internet]. 2010 Feb 10. Available from: www.mit.gov.in/sites/upload_files/dit/files/RFD_10_Feb_2010.pdf
- [34] Verma N, Verma N, Bansal R. Software development for RFD. Summer internship report no. DIT/DIR(RCM)/01. Department of Information Technology; 2010 Jun.
- [35] Chen T, Guestrin C. XGBoost: a scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2016. p. 785-94.
- [36] Kraskov A, Stögbauer H, Grassberger P. Estimating mutual information. Phys Rev E. 2004;69(6):066138.
- [37] Belghazi MI, Baratin A, Rajeswar S, Ozair S, Bengio Y, Courville A, et al. Mutual information neural estimation. In: Proceedings of the 35th International Conference on Machine Learning. 2018. p. 531-40.
- [38] Sutton RS, Barto AG. Reinforcement learning: an introduction. 2nd ed. Cambridge (MA): MIT Press; 2018.
- [39] Apache Software Foundation. Apache Spark [Internet]. Available from: <https://spark.apache.org>; Apache Kafka [Internet]. Available from: <https://kafka.apache.org>
- [40] Neo4j Inc. Neo4j graph platform [Internet]. 2023. Available from: <https://neo4j.com>
- [41] Hogan A, Blomqvist E, Cochez M, d'Amato C, de Melo G, Gutierrez C, et al. Knowledge graphs. ACM Comput Surv. 2021;54(4):71.
- [42] Microsoft Corporation. Power BI: business analytics service [Internet]. 2023. Available from: <https://powerbi.microsoft.com>

Author Profile

Tanushree Gupta works at Intuit Inc., California, USA. She developed the original information gain module in 2010 with Shefali Kapoor (Maharaja Agrasen Institute of Technology, New Delhi) as a summer internship project at the Department of Information Technology, MCIT, New Delhi [4]. The method was later reported with A. B. Patki and R. C. Meharde of that department [5].