

MODULE 2

Modeling

MODULE 2: Modeling

Textbook Reference: Ch.3 (3.1-3.6)

2.1 IR Models — Introduction & Taxonomy

An IR model is formally defined by a quadruple $[D, Q, F, R(q_i, d_j)]$ where D is the set of logical views of documents, Q is the set of logical views of queries, F is a framework for modeling document/query representations and their relationships, and R is a ranking function that assigns a real number expressing the relevance order between a query q_i and document d_j .

Models are classified along two independent dimensions:

- Mathematical basis: Set-theoretic (Boolean, fuzzy, extended Boolean), Algebraic (vector, generalized vector, LSI, neural network), Probabilistic (classic probabilistic, BM25, Bayesian networks, language models).
- Retrieval task treatment: Ad hoc retrieval (fixed collection, changing queries) vs Filtering (fixed user profile, streaming documents).

2.2 Classic Information Retrieval Models

A. Boolean Model

Based on set theory and Boolean algebra. Documents and queries are represented as sets of index terms; queries are Boolean expressions using AND, OR, NOT. A document is predicted relevant only if it exactly satisfies the Boolean expression.

- Advantages: clean formalism, simple to implement, exact/predictable semantics.
- Disadvantages: no ranking (binary decision only), exact matching may retrieve too few or too many documents, difficult for average users to build correct Boolean queries, all matched documents are considered equally relevant.

B. Vector Space Model (VSM)

Documents and the query are represented as vectors in a t -dimensional space, where t is the number of distinct index terms. Each dimension is weighted, typically using TF-IDF:

TF-IDF Weighting: $w(i,j) = tf(i,j) \times idf(i)$, where $tf(i,j)$ = frequency of term i in document j (often normalized), and $idf(i) = \log(N / df(i))$, N = total number of documents, $df(i)$ = number of documents containing term i .

Similarity between a document and the query is computed using the Cosine Similarity between their vectors:

Cosine Similarity: $\text{sim}(d_j, q) = (d_j \cdot q) / (|d_j| \times |q|)$ — the cosine of the angle between the document vector and query vector; ranges from 0 (no similarity) to 1 (identical direction).

- Advantages: partial matching allowed, produces a ranked list of documents, term-weighting improves retrieval performance over pure Boolean matching.
- Disadvantages: assumes index term independence (ignores term co-occurrence/semantics); no theoretical justification for exact weighting scheme used (heuristic).

C. Probabilistic Model

Based on the Probability Ranking Principle (PRP): documents should be ranked in decreasing order of their probability of relevance to the query, given all available evidence, to maximize overall retrieval effectiveness.

The classic model (Robertson–Sparck Jones) estimates an odds ratio of relevance using term presence in relevant vs non-relevant documents, initially guessed and refined through relevance feedback.

- Advantages: documents ranked by probability of relevance — a strong theoretical basis (PRP).
- Disadvantages: needs to guess initial separation of relevant/non-relevant documents; assumes term independence; does not use term frequency information (in the classic binary independence model).

2.3 Alternative Set-Theoretic Models

A. Fuzzy Set Model

Queries and documents are represented using fuzzy sets: each index term has a membership degree (0 to 1) in a document, instead of a binary presence/absence. This handles vagueness in term-document association, resolving Boolean model's rigidity.

B. Extended Boolean Model

Combines the flexibility of the Vector model with the precise structure of the Boolean model. Uses a p-norm (generalized distance) formulation so that AND and OR operators become 'soft' — allowing partial matches to be ranked rather than strictly included/excluded.

2.4 Alternative Algebraic Models

A. Generalized Vector Space Model (GVSM)

Removes the pairwise term-independence assumption of the classic VSM by considering the correlations between index terms — documents are represented in a space spanned by min-terms derived from term co-occurrence patterns.

B. Latent Semantic Indexing / Analysis (LSI/LSA)

Applies Singular Value Decomposition (SVD) to the term-document matrix to reduce dimensionality, mapping terms and documents into a lower-dimensional 'concept' space. This captures latent semantic relationships between terms (synonymy, polysemy) that raw term-matching misses.

SVD Decomposition: $A (t \times N) = U \cdot \Sigma \cdot V^T$ — where A is the term-document matrix, U and V are orthogonal matrices, and Σ is diagonal with singular values. Keeping only the k largest singular values gives a rank- k approximation used for retrieval in reduced dimensions.

C. Neural Network Model

Models documents, terms, and queries as nodes in a three-layer neural network; activation of query-term nodes propagates through the network (spreading activation) to activate related document nodes — capable of capturing associative/indirect relationships between terms and documents.

2.5 Alternative Probabilistic Models

A. BM25 (Okapi BM25)

The most influential and widely used probabilistic ranking function in practice, extending the classic probabilistic model with term-frequency and document-length normalization:

BM25 Formula: $\text{score}(D,Q) = \sum \text{over query terms of } \text{IDF}(q_i) \times [f(q_i,D) \times (k_1+1)] / [f(q_i,D) + k_1 \times (1 - b + b \times |D|/\text{avgdl})]$ — k_1 controls term-frequency saturation, b controls document-length normalization (typical $k_1=1.2-2.0$, $b=0.75$).

B. Bayesian Network Model

Uses a Bayesian (belief) network with two component networks — a document network and a query network — connected through index-term nodes, to compute the probability that a document satisfies the query while naturally modeling term dependencies.

C. Language Model (LM) Approach

Each document is used to estimate a unigram (or n-gram) probabilistic language model; documents are ranked by the probability that their language model would generate the query ($P(Q|M_d)$), typically with smoothing (Jelinek-Mercer or Dirichlet) to handle unseen terms.

2.6 Other / Structured Models (Overview)

- Set-based Model: represents documents as sets of terms and ranks using term co-occurrence pattern mining (association rules).
- Structured text retrieval models: combine content retrieval with document structure (covered in depth in Module 5 — XML retrieval).
- Web-specific retrieval models: incorporate link-based evidence (PageRank/HITS) alongside content-based scores (covered in Module 5).

Quick Comparison Table: Boolean = exact match, no ranking | Vector = ranked, TF-IDF + cosine | Probabilistic/BM25 = ranked by $P(\text{relevance})$ | Fuzzy/Extended Boolean = 'soft' Boolean | GVSM/LSI = handles term correlation & synonymy | Neural/Bayesian Net = associative, graph-based reasoning.

4. Worked Numerical Examples

Example 1 — Boolean Model Query Evaluation

Collection: $D_1 = \{\text{information, retrieval, web}\}$, $D_2 = \{\text{information, system}\}$, $D_3 = \{\text{web, search, system}\}$.

Query: information AND (web OR system).

- D_1 : has 'information' = true; has ('web' OR 'system') = true (web present) → D_1 satisfies the query.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

- D2: has 'information' = true; has ('web' OR 'system') = true (system present) -> D2 satisfies the query.
- D3: has 'information' = false -> D3 does NOT satisfy the query.

Result set = {D1, D2} — an unordered set with no ranking, illustrating the Boolean model's exact-match, all-or-nothing behaviour.

Example 2 — TF-IDF Weighting and Cosine Similarity

Collection of $N = 4$ documents. Term 'retrieval' occurs in $df = 2$ of them, so $idf = \log_2(4/2) = \log_2(2) = 1$. Suppose in document D , $tf('retrieval') = 3$ (raw count). Then:

Step 1 — Weight: $w('retrieval', D) = tf \times idf = 3 \times 1 = 3$

Suppose document D is represented (after weighting all terms) as vector $d = (3, 0, 2)$ over terms (retrieval, web, system), and the query $q = (1, 0, 1)$ (query contains 'retrieval' and 'system'). Then:

Step 2 — Cosine Similarity: $d \cdot q = (3 \times 1) + (0 \times 0) + (2 \times 1) = 5$ $|d| = \sqrt{3^2 + 0^2 + 2^2} = \sqrt{13} \approx 3.61$ $|q| = \sqrt{1^2 + 0^2 + 1^2} = \sqrt{2} \approx 1.41$ $\text{sim}(d, q) = 5 / (3.61 \times 1.41) \approx 5 / 5.09 \approx 0.98$

A cosine value close to 1 indicates D is highly similar in direction to the query vector, so it would be ranked near the top of the result list — unlike the Boolean model, this is a graded score usable for ranking.

Example 3 — BM25 Scoring (Conceptual Walkthrough)

For a query term q_i appearing $f = 4$ times in a document of length $|D| = 120$ words, with average document length $avgdl = 100$, and typical parameters $k_1 = 1.5$, $b = 0.75$:

BM25 term contribution: Denominator = $f + k_1 \times (1 - b + b \times |D|/avgdl) = 4 + 1.5 \times (1 - 0.75 + 0.75 \times 1.2) = 4 + 1.5 \times (0.25 + 0.9) = 4 + 1.725 = 5.725$ Numerator = $f \times (k_1 + 1) = 4 \times 2.5 = 10$ Term score (before IDF) = $10 / 5.725 \approx 1.75$, then multiplied by $IDF(q_i)$.

Notice the score grows with term frequency f but saturates (does not grow linearly forever) — this is exactly the effect of the k_1 saturation parameter, and the document being 20% longer than average slightly increases the denominator, penalising long documents slightly (length normalization via b).

5. Summary of Key Points — Module 2

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

- Every IR model is defined by $[D, Q, F, R(q_i, d_j)]$ — logical views of documents/queries, a representation framework, and a ranking function.
- Boolean model: exact match, no ranking. Vector model: ranked via TF-IDF + cosine similarity. Probabilistic model: ranked via Probability Ranking Principle.
- Fuzzy and Extended Boolean models 'soften' the rigid Boolean model to allow partial matching and ranking.
- GVSM and LSI address the Vector model's term-independence weakness; LSI uses SVD to capture latent semantic structure.
- BM25 is the practical, dominant probabilistic ranking function today, tuned by k_1 (TF saturation) and b (length normalization).
- Bayesian Networks and Language Models offer alternative, theoretically grounded probabilistic ranking approaches.

6. Glossary of Key Terms

Term	Definition
tf (term frequency)	Number of times a term occurs in a document (often normalized).
idf (inverse document frequency)	$\log(N/df)$ — down-weights terms that occur in many documents.
Cosine similarity	Cosine of the angle between document and query vectors; ranges 0 to 1.
SVD	Singular Value Decomposition — matrix factorization used by LSI for dimensionality reduction.
PRP	Probability Ranking Principle — rank documents by decreasing probability of relevance.
k_1, b (BM25)	Parameters controlling term-frequency saturation and document-length normalization.

7. Quick-Revision Question List — Module 2

- Define an IR model formally and explain the taxonomy of models.
- Explain the Boolean model with advantages/disadvantages; solve a small Boolean query example.
- Derive TF-IDF weighting and cosine similarity; solve a numerical example.
- State the Probability Ranking Principle.
- Explain the Fuzzy Set model and the Extended Boolean model.
- Explain GVSM and LSI (with SVD).

- Explain the Neural Network retrieval model.
- Explain BM25 and compute a term score given tf , document length, and parameters.
- Explain the Bayesian Network model and the Language Model approach.

8. Detailed Model Answers — Module 2 (Exam-Ready)

Q1. Define an IR model formally and explain the taxonomy of models.

An IR model is formally defined as a quadruple $[D, Q, F, R(q_i, d_j)]$. D is the set of logical representations (views) adopted for documents in the collection — for instance, a set of index terms, a weighted vector, or a probability distribution. Q is the analogous set of logical representations for user queries. F is the modeling framework: a formal structure (set theory, linear algebra, or probability theory) defining how document representations, query representations, and the relationships between them are built and manipulated. $R(q_i, d_j)$ is a ranking function that takes a query q_i and a document d_j and returns a real number quantifying their degree of match, used to order the collection. Models are classified along two orthogonal axes. By mathematical basis, models are set-theoretic (Boolean, Fuzzy, Extended Boolean — documents are sets, and matching is set membership/overlap), algebraic (Vector, Generalized Vector, LSI, Neural Network — documents are points/vectors in a space, and matching is geometric), or probabilistic (classic Probabilistic, BM25, Bayesian Network, Language Model — matching is expressed as an estimate of $P(\text{relevance})$). By retrieval task, models support ad hoc retrieval (a fixed collection is searched repeatedly with different, changing queries — the typical web-search scenario) or filtering (a fixed, standing user profile/query is applied to a stream of new incoming documents — the typical news-alert scenario).

Q2. Explain the Boolean model with advantages/disadvantages; solve a small example.

The Boolean model represents both documents and queries as sets of index terms, and queries are Boolean expressions combining terms with AND, OR, and NOT. A document is predicted relevant if and only if it satisfies the Boolean expression exactly — there is no notion of degree. Its main advantage is a clean, well-understood formalism: the semantics of a query are completely unambiguous and predictable, and the model is straightforward to implement efficiently using set intersection/union over postings lists. Its disadvantages are significant in practice: it provides no ranking at all, so all matching documents are treated as equally relevant regardless of how many query terms they actually contain or how prominently; exact matching means a single missing term (even in an OR clause structured incorrectly) can silently exclude a relevant document, or an overly broad OR can flood the user with irrelevant results; and formulating a correct Boolean query requires a level of precision (correct operator nesting, understanding

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

of AND vs OR semantics) that average users typically do not have. Worked example: for documents $D1=\{\text{info, retrieval, web}\}$, $D2=\{\text{info, system}\}$, $D3=\{\text{web, search, system}\}$ and query 'information AND (web OR system)', $D1$ and $D2$ both satisfy the query ($D1$ via 'web', $D2$ via 'system') while $D3$ fails since it lacks 'information' — giving the unordered result set $\{D1, D2\}$.

Q3. Derive TF-IDF weighting and cosine similarity; solve a numerical example.

The Vector Space Model weights each term-document pair using $w(i,j) = \text{tf}(i,j) \times \text{idf}(i)$. The term-frequency component, $\text{tf}(i,j)$, captures the intuition that a term occurring more often within a specific document is more indicative of that document's topic (often normalized to avoid unfairly favouring longer documents). The inverse-document-frequency component, $\text{idf}(i) = \log(N/\text{df}(i))$, captures the complementary intuition that a term occurring in very many documents across the collection (like common domain words) carries little discriminating power and should be down-weighted, while a term occurring in only a few documents is a strong, specific signal and should be up-weighted. Multiplying these gives a weight that rewards terms that are both locally frequent and globally rare — exactly the terms most useful for distinguishing this document from others. Once every document and the query are represented this way as vectors, similarity is computed as the cosine of the angle between them: $\text{sim}(d_j, q) = (d_j \cdot q) / (|d_j| |q|)$. This normalizes out document length (a longer document simply pointing in a similar direction is not unfairly boosted just because its raw dot product is larger), producing a score in $[0,1]$ usable directly for ranking. Numerically: with $\text{idf}(\text{'retrieval'}) = \log_2(4/2) = 1$ and $\text{tf} = 3$, $\text{weight} = 3$; for $d = (3,0,2)$ and $q = (1,0,1)$, $d \cdot q = 5$, $|d| \approx 3.61$, $|q| \approx 1.41$, giving $\text{sim} \approx 0.98$ — a strong match.

Q4. State the Probability Ranking Principle.

The Probability Ranking Principle (PRP), the theoretical foundation of the classic probabilistic model, states: if an IR system's response to a query is a ranking of documents in decreasing order of their estimated probability of relevance to that query — given all evidence available to the system — then the overall effectiveness of the system to its users will be the best obtainable from that evidence. In other words, ranking purely by $P(\text{relevant} \mid \text{document, query})$ is provably optimal under reasonable assumptions about how users interact with a ranked list. The classic probabilistic model operationalizes this using term-presence patterns: it estimates, for each query term, the odds that a document containing that term is relevant versus non-relevant, initially guessed (often assuming a uniform or simple prior) and then refined as relevance judgments become available through feedback, iteratively converging toward a better-separated ranking of relevant versus non-relevant documents.

Q5. Explain the Fuzzy Set model and the Extended Boolean model.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

The Fuzzy Set model directly targets the Boolean model's core weakness — its rigid, binary in/out-of-set membership — by allowing each index term to have a graded membership degree (a real number between 0 and 1, rather than strictly 0 or 1) in a document, computed from term co-occurrence patterns or weighting schemes. Boolean operators are then redefined using fuzzy set operations (e.g., AND as the minimum of memberships, OR as the maximum), which allows the model to express degrees of match and produce a ranking, while still keeping the familiar, structured Boolean query syntax. The Extended Boolean model takes a related but distinct approach: it embeds Boolean queries into the geometric structure of the Vector model using a p-norm formulation, a generalized distance measure parameterized by p. As p varies, the AND and OR operators smoothly interpolate between strict Boolean semantics (at one extreme) and pure Vector-model averaging (at the other), letting the query author tune how 'strict' or 'soft' the Boolean structure should be while still obtaining a ranked, graded output rather than a strict in/out decision.

Q6. Explain GVSM and LSI (with SVD).

The classic Vector model assumes index terms are pairwise orthogonal (mutually independent) dimensions — an assumption known to be false in real text, where terms co-occur and relate semantically. The Generalized Vector Space Model (GVSM) relaxes this by constructing a space based on minterms (patterns of term co-occurrence across the collection) rather than raw terms, so correlations between terms are implicitly represented in the geometry of the space, allowing documents sharing correlated (though not identical) vocabulary to be judged more similar than the classic model would allow. Latent Semantic Indexing (LSI) tackles the same underlying problem — synonymy (different words, same meaning) and polysemy (same word, different meanings) — using a different mathematical tool: Singular Value Decomposition. The term-document matrix A (size $t \times N$) is factored as $A = U \cdot \Sigma \cdot V^T$, where U and V are orthogonal matrices capturing term and document 'concept' loadings respectively, and Σ is a diagonal matrix of singular values ordered by importance. By keeping only the k largest singular values (and the corresponding rows/columns of U and V), LSI produces a rank- k approximation of A that represents documents and terms in a much lower-dimensional 'latent semantic' space, where documents about the same underlying concept cluster together even if they share few literal terms — directly mitigating both synonymy and polysemy issues that plague literal term matching.

Q7. Explain the Neural Network retrieval model.

The Neural Network model represents queries, documents, and index terms as nodes in a (typically three-layer) network — a query-term layer, an index-term layer, and a document layer — connected by weighted edges reflecting term-document association strength (e.g., derived from tf-idf weights). Retrieval proceeds via spreading activation: query terms are activated first, and this activation propagates along the edges to

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

activate associated index terms, which in turn propagate activation onward to activate the documents connected to them. Because activation can flow through intermediate, indirectly-connected nodes (not just terms literally present in the query), the model is capable of retrieving documents connected to the query through associative or indirect relationships that a literal term-matching model would miss entirely — a conceptual precursor to the associative, embedding-based retrieval methods used in modern neural IR.

Q8. Explain BM25 and compute a term score.

BM25 (Okapi BM25) is the most influential and widely deployed practical ranking function derived from the probabilistic model family. It refines the classic model by explicitly incorporating term frequency (how often a query term appears in a document) and document-length normalization (penalizing documents that are unusually long relative to the collection average, since long documents naturally accumulate higher raw term counts without necessarily being more relevant). Its score for a query term combines $IDF(q_i)$ with a saturating term-frequency component: $f(q_i, D) \times (k_1 + 1)$ divided by $f(q_i, D) + k_1 \times (1 - b + b \times |D|/avgdl)$. The parameter k_1 controls how quickly additional occurrences of a term stop adding much extra score (saturation — a document mentioning a term 20 times is not treated as vastly more relevant than one mentioning it 5 times), while b (between 0 and 1) controls the strength of length normalization ($b=0$ disables it entirely; $b=1$ applies full normalization). Worked example: for $f=4$, $|D|=120$, $avgdl=100$, $k_1=1.5$, $b=0.75$: denominator = $4 + 1.5 \times (1 - 0.75 + 0.75 \times 1.2) = 4 + 1.725 = 5.725$; numerator = $4 \times 2.5 = 10$; term score (pre-IDF) ≈ 1.75 — demonstrating both the saturation and the mild length penalty in action.

Q9. Explain the Bayesian Network model and the Language Model approach.

The Bayesian Network model expresses retrieval as inference over a belief network with two connected component networks — a document network (modeling how documents generate/support index terms) and a query network (modeling how the query is composed from index terms) — joined at shared index-term nodes. This structure allows the model to naturally represent dependencies between terms (unlike the classic probabilistic model's independence assumption) and to combine multiple sources of evidence for relevance within a single coherent probabilistic framework, computing $P(\text{relevance} \mid \text{document}, \text{query})$ via standard belief-propagation inference. The Language Model (LM) approach takes a different but equally probabilistic route: rather than modeling relevance directly, it treats each document as having generated a probabilistic language model (typically a unigram model — a simple probability distribution over vocabulary words, though n-gram extensions exist), and ranks documents by the probability that document D 's language model would generate the observed query Q , i.e., $P(Q \mid M_d)$. Since a document's raw word counts alone would assign zero probability to any query term absent from that specific document (an overly harsh penalty), smoothing techniques such as Jelinek-Mercer interpolation or Dirichlet-prior

smoothing blend the document's own statistics with corpus-wide statistics, ensuring even unseen query terms receive a small, non-zero, principled probability rather than eliminating the document outright.

9. Real-World Application — Model Choice in Practice

Modern commercial search and enterprise search systems rarely use a single 'pure' model from the taxonomy above; instead they typically use BM25 (or a close variant) as a fast, effective first-stage retrieval/ranking function over the inverted index — because it is cheap to compute, robust across query types, and requires no training data — and then apply a more expensive, learned re-ranking model only to the small number of top candidates BM25 returns. LSI-style dimensionality reduction ideas live on in modern dense/embedding-based retrieval (which similarly maps text into a lower-dimensional latent space to catch synonymy), while the Boolean model persists in specialized settings such as legal and patent search, where lawyers deliberately want exact, auditable, reproducible result sets rather than an opaque ranked list — illustrating that model choice depends heavily on whether the task values recall-with-precision-ranking (web search) or perfect, exact reproducibility (legal discovery).

10. Common Mistakes & Exam Tips — Module 2

- Do not describe the Vector model as having 'no theoretical basis' without qualification — it lacks a theoretical justification specifically for its weighting scheme (TF-IDF is heuristic), not for the geometric ranking approach itself.
- When explaining BM25, always name both k_1 and b explicitly and state what each controls — answers that give only the formula without explaining the parameters lose significant marks.
- Do not confuse GVSM (handles term correlation within the Vector model framework) with LSI (uses SVD for dimensionality reduction) — they solve a related problem (term independence/synonymy) using different mathematical machinery, and examiners often ask you to distinguish them explicitly.
- For the PRP, state it as a ranking principle (ordering documents by $P(\text{relevance})$), not merely as 'the probabilistic model uses probability' — the exact principle statement is what earns marks.

11. Additional Solved Practice Problems — Module 2

Problem 1: Boolean query with NOT

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

Collection: $D1=\{\text{data},\text{retrieval},\text{web}\}$, $D2=\{\text{data},\text{mining}\}$, $D3=\{\text{web},\text{mining},\text{search}\}$. Evaluate the query: data AND NOT mining.

Solution: D1: has 'data'=true, has 'mining'=false, so NOT mining=true $\rightarrow$ D1 satisfies query. D2: has 'data'=true, has 'mining'=true, so NOT mining=false $\rightarrow$ D2 fails. D3: has 'data'=false $\rightarrow$ fails. Result = {D1}.

Problem 2: TF-IDF computation from scratch

A collection has $N=8$ documents. The term 'index' appears in 4 of them ($df=4$). In document D, 'index' occurs 6 times ($tf=6$, unnormalized). Compute the TF-IDF weight.

Solution: $idf = \log_2(N/df) = \log_2(8/4) = \log_2(2) = 1$. Weight = $tf \times idf = 6 \times 1 = 6$.

Problem 3: Cosine similarity ranking

Given query vector $q=(1,1,0)$ and two documents $d1=(2,2,0)$ and $d2=(0,0,3)$ over the same 3 terms, which document should rank higher and why?

Solution: $\text{sim}(d1,q) = (2+2+0)/(\sqrt{8}\sqrt{2}) = 4/4 = 1.0$ ($d1$ points in EXACTLY the same direction as q). $\text{sim}(d2,q) = (0+0+0)/(3\sqrt{2}) = 0$ (no shared non-zero terms at all). $d1$ ranks higher — despite $d2$ possibly being a 'longer' or 'denser' document, cosine similarity correctly ignores magnitude and rewards direction/term-overlap with the query, which $d2$ has none of.

Problem 4: BM25 parameter effect

For a fixed term frequency f and IDF, explain (without recomputation) what happens to a document's BM25 score if that document's length $|D|$ is much greater than avgdl , versus a document whose length equals avgdl exactly.

Solution: When $|D| \gg \text{avgdl}$, the term $(|D|/\text{avgdl})$ in the denominator becomes large, inflating the denominator and therefore LOWERING the BM25 term score relative to a document of average length with the identical term frequency — this is the length-normalization effect controlled by b , penalizing documents that are long simply because they are long, not because they are more relevant.

Problem 5: Model selection

A legal team needs to retrieve EVERY document mentioning a specific contract clause, with a fully reproducible, auditable result set for court proceedings — no ranking or 'approximate' matches are acceptable. Which classic IR model is most appropriate, and why?

Solution: The Boolean model is most appropriate here. Its exact-match, all-or-nothing semantics produce a precisely reproducible, auditable result set (the same query always returns exactly the same documents), which is exactly what a legal discovery process requires — unlike the Vector or Probabilistic models, whose ranked, probabilistic output is not naturally suited to an 'exhaustive and exact' requirement.

12. Algorithm/Process Reference — Module 2

Pseudocode: Vector Space Model Ranking

Pseudocode: function vectorRank(query, documents): qVec = buildTFIDFVector(query) scores = [] for doc in documents: dVec = buildTFIDFVector(doc) // precomputed at index time score = dotProduct(qVec, dVec) / (norm(qVec) * norm(dVec)) scores.append((doc, score)) return sortDescending(scores)

Pseudocode: BM25 Scoring

Pseudocode: function bm25Score(query, doc, avgdl, k1, b): score = 0 for term in query: f = termFrequency(term, doc) idf = computeIDF(term) numerator = f * (k1 + 1) denominator = f + k1 * (1 - b + b * len(doc)/avgdl) score += idf * (numerator / denominator) return score

13. Historical Context & Evolution of IR Models

The Boolean model dominated the first decades of computerized IR (1960s-1970s) simply because it was the most natural fit for the exact, deterministic computing hardware and small, specialist user base of the time. Salton's introduction of the Vector Space Model in the late 1960s/1970s (through the SMART retrieval system) was a turning point, proving that a purely heuristic, ranking-based approach could substantially outperform exact Boolean matching for real users. The classic Probabilistic model (Robertson-Sparck Jones, 1970s-80s) provided the missing theoretical justification — the Probability Ranking Principle — for WHY ranking is optimal, which the Vector model had lacked. BM25, developed through the 1990s Okapi project, then translated this theory into the practically dominant ranking function still used (often as a strong first-stage baseline) in modern search infrastructure today. LSI (1990s) and later neural/embedding-based retrieval methods represent the ongoing effort to solve the vocabulary-mismatch problem (synonymy/polysemy) that pure term-matching approaches, however well-weighted, cannot solve by counting words alone.

14. Frequently Asked Questions — Module 2

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

- Q: Why is cosine similarity preferred over raw dot product for the Vector model? A: The dot product alone is biased toward longer documents (which naturally accumulate larger term-weight sums); dividing by the vector magnitudes normalizes this away, so similarity reflects directional/topical alignment rather than document length.
- Q: Does BM25 completely replace the classic probabilistic model? A: No — BM25 IS a probabilistic model, specifically a refinement of the classic model's scoring function with added term-frequency saturation and length normalization; it did not abandon the underlying probabilistic framework.
- Q: Is LSI still used today? A: Its core idea (dimensionality reduction to capture latent semantic structure) persists conceptually in modern dense/embedding retrieval methods, though LSI's specific SVD-based implementation has mostly been superseded by neural embedding models.

15. Self-Assessment Checklist — Module 2

- I can state the $[D, Q, F, R]$ formal definition of an IR model and both classification axes.
- I can explain the Boolean model's mechanics and evaluate a sample Boolean query by hand.
- I can derive TF-IDF and compute cosine similarity for a small numerical example.
- I can state the Probability Ranking Principle precisely.
- I can explain how the Fuzzy Set and Extended Boolean models 'soften' Boolean matching.
- I can explain GVSM and LSI, including the role of SVD.
- I can write the BM25 formula and explain the role of k_1 and b .
- I can explain the Bayesian Network and Language Model approaches and how they differ.

16. Multiple Choice Review Questions — Module 2

- Q1. In $[D, Q, F, R(q_i, d_j)]$, what does R represent? (a) document representation (b) ranking function (c) query language (d) index structure. Answer: (b) — $R(q_i, d_j)$ is the ranking function producing the relevance score.
- Q2. The Boolean model's biggest practical weakness is: (a) too slow (b) no ranking of results (c) cannot use AND/OR (d) requires GPUs. Answer: (b) — all matches are treated as equally relevant.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

- Q3. $\text{idf}(i) = \log(N/\text{df}(i))$ down-weights terms that: (a) never appear (b) appear in very few documents (c) appear in very many documents (d) are stopwords only. Answer: (c) — common terms across the collection carry less discriminating power.
- Q4. Cosine similarity ranges over: (a) -1 to 1 (b) 0 to 1 for non-negative vectors (c) 0 to infinity (d) -infinity to infinity. Answer: (b) — TF-IDF weights are non-negative, so cosine similarity is confined to [0,1].
- Q5. The Probability Ranking Principle states documents should be ranked by: (a) alphabetical order (b) decreasing probability of relevance (c) document length (d) publication date. Answer: (b).
- Q6. LSI uses which mathematical technique? (a) Fourier Transform (b) Singular Value Decomposition (c) Huffman coding (d) Dynamic programming. Answer: (b) SVD.
- Q7. In BM25, increasing parameter b (toward 1) has what effect? (a) disables length normalization (b) increases length normalization strength (c) increases term-frequency saturation (d) has no effect. Answer: (b) — b controls length normalization strength; $b=0$ disables it entirely.
- Q8. The Extended Boolean model's p -norm formulation interpolates between: (a) Boolean and Probabilistic models (b) Boolean and Vector models (c) LSI and BM25 (d) Fuzzy and Neural models. Answer: (b).
- Q9. Which model is most naturally suited to legal/patent search requiring exact, reproducible results? (a) Vector model (b) BM25 (c) Boolean model (d) LSI. Answer: (c) Boolean model, for its exact, deterministic semantics.

17. Extended Case Study — Choosing a Ranking Function for a Startup Search Product

A startup building a document-search product for enterprise customers must choose an initial ranking approach. Pure Boolean matching is rejected early because business users cannot reliably write correct Boolean queries and expect a ranked, 'Google-like' experience. Between the Vector model and BM25, the team chooses BM25, since it retains the practical simplicity and speed of TF-IDF-style scoring (no training data required, unlike a fully learned ranker) while its term-frequency saturation and length normalization measurably outperform naive cosine similarity on their internal benchmark of long, unevenly-sized documents (some 200 words, some 20,000). They defer GVSM/LSI-style semantic matching to a later phase, since it requires more engineering investment (building and maintaining the SVD-reduced representation as the collection grows) for a benefit that matters more once vocabulary-mismatch

complaints start appearing in user feedback — illustrating a realistic, staged model-adoption decision grounded directly in the trade-offs this module covers.

18. Cross-Module Connections

The IR models introduced here are the concrete 'Ranking' block from Module 1's system diagram. The evaluation metrics of Module 3 (Precision, Recall, MAP) are precisely how these different models get compared and validated against each other. BM25 reappears directly in Module 5 as the typical content-relevance component combined with PageRank/HITS link-based signals in web ranking, and the inverted index of Module 4 is the physical data structure that makes all of these models' postings-based computations (term lookups, Boolean intersection, TF-IDF accumulation) efficient at scale.

19. Diagram Walkthroughs — What to Draw and How

Diagram 1: Vector Space Model Geometry (step-by-step drawing guide)

- Step 1: Draw two axes (for a simplified 2-term example), labeled with two index terms, e.g., 'retrieval' (x-axis) and 'web' (y-axis).
- Step 2: Plot the query vector q as an arrow from the origin to a point, e.g., (1,1), and label it 'Query'.
- Step 3: Plot two or three document vectors as arrows from the origin to their respective (tf-idf weighted) coordinates, labeling each 'D1', 'D2', etc.
- Step 4: Draw the angle θ between the query vector and each document vector — SMALLER angle means HIGHER cosine similarity; label this clearly, since this is the core visual insight examiners look for.
- Step 5: Rank the documents by increasing angle (decreasing cosine similarity) and state this ranking explicitly beside the diagram.

Figure 3.1 — Vector Space Model: Query vs Documents

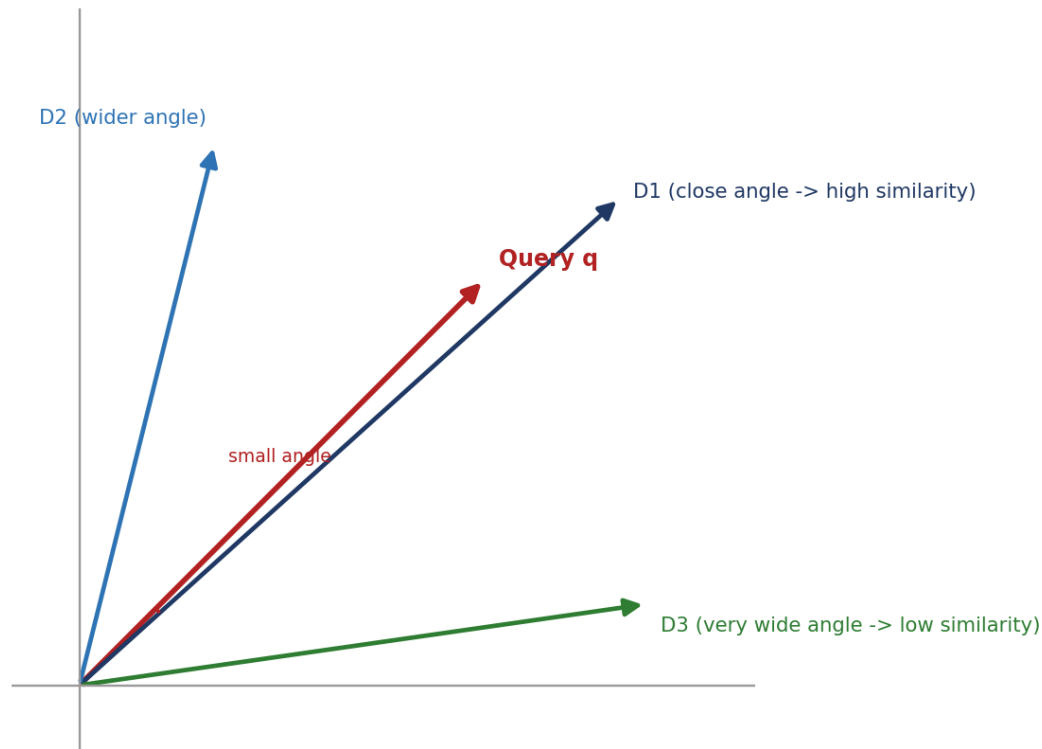


Figure 3.1 — Vector Space Model geometry: query and document vectors compared by angle.

Diagram 2: Model Taxonomy Tree (step-by-step drawing guide)

- Step 1: Draw a root node labeled 'IR Models'.
- Step 2: Draw three child branches labeled 'Set-Theoretic', 'Algebraic', and 'Probabilistic'.
- Step 3: Under Set-Theoretic, add leaves: 'Boolean', 'Fuzzy Set', 'Extended Boolean'.
- Step 4: Under Algebraic, add leaves: 'Vector (VSM)', 'GVSM', 'LSI', 'Neural Network'.
- Step 5: Under Probabilistic, add leaves: 'Classic Probabilistic', 'BM25', 'Bayesian Network', 'Language Model'.
- Step 6: Optionally annotate each leaf with one keyword (e.g., BM25 -> 'k1, b parameters') to make the tree double as a revision aid.

Figure 3.2 — Taxonomy of Information Retrieval Models

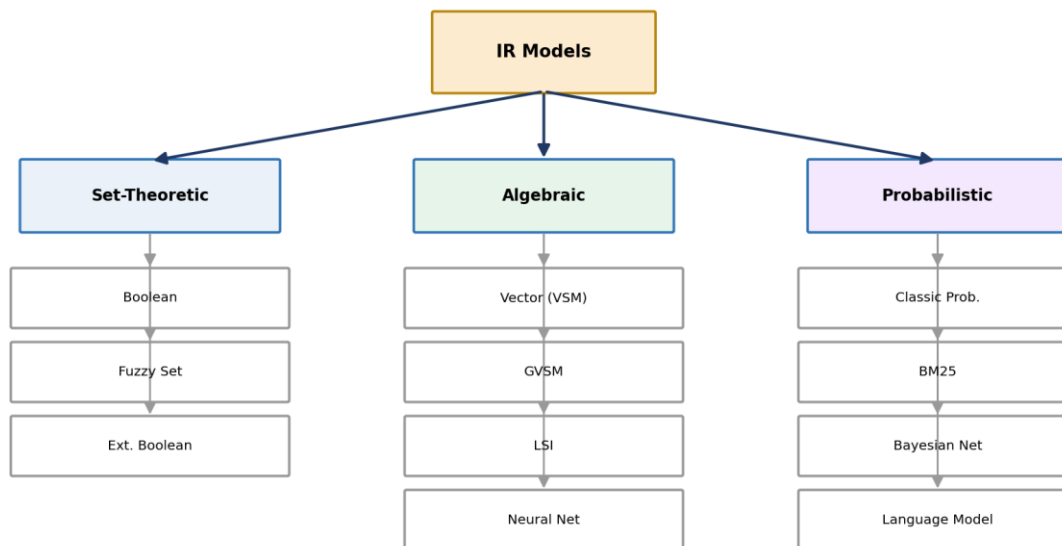


Figure 3.2 — Complete taxonomy tree of IR models covered in this module.

20. Unsolved Practice Problems (Self-Study, with Hints)

- 1. For documents $D1=\{a:2,b:1\}$, $D2=\{a:0,b:3\}$ and query $q=\{a:1,b:1\}$, compute cosine similarity for both and determine which ranks higher. (Hint: normalize by vector magnitude.)
- 2. Explain, in your own words, why the Boolean model cannot express 'this document is somewhat about topic X.' (Hint: think about set membership being binary.)
- 3. If $df('the')=10,000$ out of $N=10,000$ documents, what is $idf('the')$? What does this numerically confirm about stopwords? (Hint: $\log(N/df)$ when $df=N$.)
- 4. Explain why GVSM is described as addressing a weakness of the classic Vector model specifically, not of the Boolean model.
- 5. For BM25 with $k1=0$ (an extreme case), what happens to the term-frequency component of the score? (Hint: substitute $k1=0$ into the formula and simplify.)
- 6. Explain in one sentence each: what problem does LSI solve, and what problem does BM25 solve? Are these the same problem?
- 7. A Bayesian Network model and a Language Model approach are both 'probabilistic', yet structurally very different. Name one structural difference.

- 8. Design a simple query and a 3-document collection where the Boolean model and the Vector model would return DIFFERENT top results, and explain why.

21. Full Sample 20-Mark Exam Answer — Module 2

Question: 'Compare the Boolean, Vector, and Probabilistic classic IR models on matching type, ranking capability, and theoretical basis, and explain how BM25 extends this family.' (20 marks)

21. Full Sample 20-Mark Exam Answer — Module 2

Question: 'Compare the Boolean, Vector, and Probabilistic classic IR models on matching type, ranking capability, and theoretical basis, and explain how BM25 extends this family.' (20 marks)

Model Answer: The three classic IR models differ fundamentally along three dimensions: how they match documents to queries, whether they produce a ranking, and what theoretical justification underlies their approach. The Boolean model represents documents and queries as sets of index terms and evaluates queries as Boolean expressions (AND, OR, NOT); its matching is exact — a document either satisfies the expression or it does not — so it produces NO ranking at all, treating every matching document as equally relevant regardless of how strongly it matches. Its theoretical basis is simply classical set theory and Boolean algebra, which gives it clean, predictable semantics but no mechanism for expressing degrees of relevance. The Vector Space Model instead represents documents and the query as vectors in a t -dimensional space (one dimension per index term), typically weighted using TF-IDF, and computes similarity via the cosine of the angle between vectors; this produces a genuinely ranked, graded output, since cosine similarity yields a continuous score in $[0,1]$ rather than a binary decision. However, the Vector model's theoretical basis is heuristic rather than principled — there is no formal proof that TF-IDF weighting or cosine similarity is 'optimal' in any sense, it simply performs well empirically. The Probabilistic model closes this theoretical gap: grounded in the Probability Ranking Principle, it ranks documents by their estimated probability of relevance given the available evidence, which is PROVABLY optimal for overall retrieval effectiveness under reasonable assumptions — giving it the strongest theoretical foundation of the three, at the cost of needing to estimate probabilities (often starting from a rough initial guess refined through feedback) rather than computing a value directly from term counts. BM25 (Okapi BM25) is the modern, practically dominant extension of the probabilistic family: it retains the probabilistic ranking philosophy but adds two crucial refinements missing from the classic probabilistic model — a term-frequency saturation component (controlled by parameter k_1 , ensuring that additional occurrences of a

term contribute diminishing extra score rather than growing the score unboundedly) and a document-length normalization component (controlled by parameter b , penalizing documents that are unusually long relative to the collection average). This combination — probabilistic grounding plus practical term-frequency and length handling — is precisely why BM25 remains the standard first-stage ranking function in production search systems today, effectively inheriting the Probabilistic model's theoretical rigor while directly addressing the practical weaknesses that the classic probabilistic and Vector models each had in isolation.

22. Additional Worked Problem — Retrieval Model Selection Across Domains

Scenario: a single organization operates three different search products — a public product-search box (millions of casual users), an internal legal-hold document search (a handful of paralegals needing exhaustive, reproducible results), and a research-paper recommendation feed (personalized, based on reading history). Walk through the model choice for each. For the public product search, BM25 (or a learned re-ranker built atop BM25 candidates) is appropriate: it needs no training data to bootstrap, handles the huge, noisy vocabulary of casual queries robustly, and its ranked output matches how casual users expect results to be ordered. For the legal-hold search, the Boolean model remains the right choice specifically BECAUSE it lacks ranking — paralegals need a complete, exact, reproducible result set for every run of the same query (for audit and discovery purposes), and a probabilistic or vector-based ranking (which could vary subtly between runs if the underlying model or index changes) is actively undesirable in this context. For the research-paper recommendation feed, an LSI-style or modern embedding-based latent-semantic approach is best suited, since the goal is surfacing papers RELATED in underlying topic/concept even when they share little literal vocabulary with a user's reading history — precisely the synonymy/polysemy problem that dimensionality-reduction techniques like LSI were designed to address, and that pure term-matching models (Boolean, raw Vector, BM25) handle poorly by comparison. This scenario reinforces that 'best model' is always relative to the specific retrieval task's requirements around ranking, reproducibility, and semantic matching, not a single universally superior choice.

23. Comprehensive Quick-Reference Sheet — Module 2

Concept	Formula / One-Line Reference
IR Model	$[D, Q, F, R(q_i, d_j)]$ — doc/query views, framework, ranking function
TF-IDF weight	$w(i, j) = \text{tf}(i, j) \times \text{idf}(i), \text{idf}(i) = \log(N/\text{df}(i))$

Concept	Formula / One-Line Reference
Cosine similarity	$\text{sim}(d_j, q) = (d_j \cdot q) / (d_j \times q)$
Boolean model	Exact match via AND/OR/NOT; no ranking
PRP	Rank documents by decreasing $P(\text{relevance} \mid \text{evidence})$
Fuzzy Set model	Graded (0-1) term membership instead of binary Boolean
Extended Boolean	p-norm interpolation between Boolean and Vector semantics
GVSM	Vector model generalized using term-correlation minterms
LSI	$A (txN) = U \cdot \Sigma \cdot V^T$; keep top-k singular values for latent space
Neural Net model	Spreading activation across query-term/index-term/document layers
BM25	$\text{score} = \text{IDF} \times [f(k_1+1)] / [f + k_1(1-b+b D /\text{avgdl})]$
Bayesian Network	Document network + query network joined at index-term nodes
Language Model	Rank by $P(Q \mid M_d)$, document's own generative language model

24. Chapter Summary Essay — Module 2

Module 2 is the mathematical core of the course, and the three-way taxonomy — set-theoretic, algebraic, and probabilistic — provides a map for organizing what could otherwise feel like a long, disconnected list of models. Every model in this module is ultimately trying to solve the same underlying problem stated in Module 1's IR problem definition: estimating relevance between a query and a document, using the $[D, Q, F, R]$ formalism as a common language to compare them. The progression within the module also tells a historical and conceptual story: the Boolean model's rigid exactness gives way to the Vector model's ranked flexibility, which lacks theoretical grounding that the Probabilistic model then supplies via the Probability Ranking Principle; the Fuzzy and Extended Boolean models attempt to have Boolean's precision alongside ranking; GVSM and LSI address the Vector model's blindness to term correlation and synonymy; and BM25, the Bayesian Network model, and Language Models represent increasingly sophisticated ways of operationalizing the probabilistic ranking philosophy in practice. This module connects forward to Module 3, where the metrics needed to actually COMPARE these models empirically (Precision, Recall, MAP) are introduced, and to Module 4, where the inverted index data structure that makes any of these models computationally efficient is built, and to Module 5, where BM25-style content scoring is combined with the link-based PageRank/HITS signals unique to the Web.

25. Final Revision Mnemonic — Module 2

A useful memory aid for the model taxonomy: 'SAP' — Set-theoretic (Boolean, Fuzzy, Extended Boolean), Algebraic (Vector, GVSM, LSI, Neural Net), Probabilistic (classic Probabilistic, BM25, Bayesian Network, Language Model) — three families, each solving the same relevance-ranking problem with a different mathematical toolset. For BM25 specifically, remember 'k1 = kounts saturate, b = big documents get

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

penalized' as a quick way to keep the two parameters and their roles straight under exam pressure. For distinguishing GVSM from LSI, remember: GVSM correlates TERMS directly (minterns), while LSI reduces DIMENSIONS via SVD to find latent concepts — 'GVSM stays in term-space, LSI moves to concept-space.'