Information retrieval: the lexical foundation

SciFact

LECTURE 19 LECTURE NOTES

Applications of Machine Learning

BSc Mathematics of Artificial Intelligence · Third year

Part V begins

Part	Lectures	Source
I–IV · the textbook	1–18	Géron, Chapters 1–15
V · Search and recommendation	**19–22**	**these lecture notes**

Where we are. Last lecture built attention — queries, keys and values, scaled by $\sqrt{d_k}$ — fine-tuned a pretrained transformer as a classifier, and used one vector per review to search a corpus by meaning. Search was a side use there. From today it is the task.

> **RETRIEVAL IS NOT PREDICTION**
>
> The input is a **query** and a **corpus**; the output is a **ranking** of the corpus. Several documents may be right, their order matters, and the corpus is too large to score with a neural network for every query. Outside the book, and examined on the same terms as everything else.

Today

1. The retrieval problem, and the corpus
2. **The method** — the inverted index, term weighting, and BM25
3. **The mathematics** — evaluating a ranking: P@k, R@k, RR, AP and NDCG
4. Where the judgements come from — what a relevance judgement is, how SciFact's were made, what pooling misses, and graded relevance
5. **Design exercise** — a legal-tech company's search product for law firms, in nine steps, with the pitfalls and how to address them

Open Lecture 19 in Colab

The task

A **claim** arrives. Find the scientific abstracts that bear on it.

- “0-dimensional biomaterials lack inductive properties.”
- “Breast cancer development is determined exclusively by genetic factors.”

Nobody wants a yes or no. They want the *evidence*, ranked, so a human can read the top few and decide. That is the whole shape of the problem, and it is why the output is an ordering.

> **WHY THIS CORPUS**
>
> SciFact is small enough that we can score **every** document for **every** query exactly, with no approximate index in the way. Every number in this lecture and the next is therefore a property of the method, not of an index we did not examine.

The shape of every search system

1 Retrieve

millions → hundreds. Must be cheap.

2 Re-rank

hundreds → tens. Can be expensive.

3 Present

tens → ten. Diversity, freshness, policy.

Today is stage 1, and it is stage 1 that decides the ceiling: **a document the first stage never returns cannot be recovered by any later stage**. Stage 2 is Lecture 20. Stage 3 is mostly not machine learning, and mostly not in this course.

This is why we will keep looking at recall@100 for a system whose output is ten results. The first stage is not judged on what the user sees; it is judged on what it makes possible.

Attention falls off a results page

Whatever we build, its output is read by a person, and that person's attention is not uniform:

- The first result is examined far more often than the second
- Almost nobody reaches the tenth
- The second page effectively does not exist

> **CONSEQUENCE FOR THE MATHEMATICS**
>
> A metric that treats ranks 1 and 10 alike is **measuring something nobody experiences**. Every metric in this lecture's derivation is an attempt to encode that decay — and the differences between them are differences of opinion about its shape.

Nothing in this lecture measures user attention: it is an assumption we import, and I am flagging it as one.

The corpus, counted

Quantity	SciFact
Abstracts	5,183
Claims with judgements (test)	300
Mean abstract length, in tokens	225.2
Median abstract length	213
Vocabulary	35,734
Relevant documents per claim, mean	1.13
Claims with exactly one relevant document	92.3%

That last row matters more than it looks. With one relevant document per claim most of the time, **recall@10 and the reciprocal rank are measuring nearly the same thing** — and on a corpus where each query had twenty relevant documents they would not be.

What a document looks like

Microstructural development of human newborn cerebral white matter assessed in vivo by diffusion tensor magnetic resonance imaging.

> Alterations of the architecture of cerebral white matter in the developing human brain can affect cortical development and result in functional disabilities. A line scan diffusion-weighted magnetic resonance imaging (MRI) sequence with diffusion tensor analysis was applied to measure the apparent diffusion coefficient, to calculate …

311 tokens of title and abstract, and nothing else — no full text, no citations, no authors. Whatever a system knows about this paper, it knows from this.

Notice how it is written: “diffusion-weighted magnetic resonance imaging”, not “MRI scan”. A claim written by someone else about the same result may share very few of these words, and that observation will return, with a number attached, before this lecture ends.

The baseline: return the corpus unranked

Every number needs a baseline, here as anywhere. Before any method, what does *no* method score? Return every document in corpus order, the same order for every query:

Metric	Fixed order
Recall@10	✗ 0.3%
Reciprocal rank	✗ 0.0017
NDCG@10	✗ 0.0012

Effectively zero, and that is the point: unlike accuracy under imbalance in Lecture 3, a ranking metric has an honest floor near zero. **Any** number you report is above the baseline, so “better than nothing” is not an argument here — the comparison that matters is against the *previous method*.

First: what is a term?

Before any weighting, a decision nobody writes up: how the text becomes a list of things to count. On a real claim from our corpus:

> “0-dimensional biomaterials show inductive properties.”

`0` `dimensional` `biomaterials` `show` `inductive` `properties`

6 tokens, lower-cased, split on non-letters. Which is already three decisions — and `0-dimensional` has just become `0` and `dimensional`, two terms that mean nothing apart.

The obvious first idea: Boolean retrieval

Return the documents that contain *all* the query terms. It is exact, it is fast with an inverted index, and it was how search worked for thirty years.

$$\text{result} = \bigcap_{t \in q} \text{postings}(t)$$

Over all 300 test claims, **✗ 98.0%** match no abstract at all: an **empty page**, not a bad ranking. One word of the claim missing from an otherwise perfect abstract excludes it, and the failure is silent — an empty result looks like “no such document exists”.

> **THE LESSON THAT REORGANISES EVERYTHING**
>
> Stop **filtering** and start **scoring**. Every document gets a number, the user gets the largest, and a missing term costs something rather than everything.

Why retrieval is cheap

Scoring 5,183 documents per query by reading each one is hopeless at scale. The trick is 50 years old and it is a data structure, not a model.

> **THE INVERTED INDEX**
>
> For every *term*, store the list of documents containing it, with the count. A query of four words then touches only the documents containing at least one of those four — not the corpus.

Vocabulary (distinct terms)	35,734
Postings (term–document pairs)	633,514
Full matrix, were it dense	185 million cells

The postings list is **0.3%** of the dense matrix. Sparsity is not an optimisation here; it is the reason the whole field is possible.

Term frequency, and why it is not enough

A document mentioning “biomaterials” five times is probably more about biomaterials than one mentioning it once. So score by how often the query terms occur.

> **TWO FAILURES, IMMEDIATELY**
>
> **Common words dominate.** Every abstract contains “of” and “the”, so a query containing them matches everything equally.
>
> **Long documents win.** A 900-word abstract contains more of everything than a 100-word one, and is not thereby more relevant.

The rest of the method is two corrections, one for each failure — and each has a parameter, so each can be measured.

Saturation, and the parameter k_1

The first correction to raw counting: the tenth occurrence of a word should not count ten times the first.

$$\frac{\mathrm{tf}\,(k_1+1)}{\mathrm{tf}+k_1} \longrightarrow k_1+1 \ \text{ as } \mathrm{tf}\to\infty$$

tf	1	2	3	5	10	20
factor	1.000	1.310	1.462	1.610	1.743	1.818

From 1 to 2 occurrences the factor rises 0.310; from 10 to 20 it rises 0.075. The curve is bounded, so no single word can dominate a score by repetition — which is also, incidentally, the first anti-spam device in search.

Choosing k_1 by measurement

k_1 sets how fast the saturation bites: small k_1 saturates almost immediately, large k_1 approaches raw counting. Measured over 150 claims, b held at 0.4:

k_1	0.5	0.9	1.2	2.0
NDCG@10	0.6883	0.6937	0.6944	0.6926

The spread across a fourfold change in k_1 is 0.0061. Which is the honest finding, and not the one you would guess from how much has been written about tuning it: **on abstracts, this parameter barely matters**.

It matters on long documents, where a term can occur fifty times. Report the corpus with the parameter, always.

Inverse document frequency

$$\mathrm{idf}(t) = \log\left(1 + \frac{N - n_t + 0.5}{n_t + 0.5}\right)$$

N DOCUMENTS, N_T OF THEM CONTAINING T

Term	Documents	idf
`of`	5,173	0.0020
`the`	5,159	0.0047
`and`	5,158	0.0049

`of` appears in 5,173 of 5,183 abstracts and earns a weight of 0.0020. **No stop-word list was needed**: the weighting derives the stop words from the corpus it is given.

Stop words, derived rather than declared

Classical systems shipped a hand-written list of words to delete: *the, of, and, in, to* … It works, and it is a liability:

- The list is language-specific, and someone must maintain it
- It is corpus-blind — in a corpus about the band The Who, deleting *the* and *who* destroys the query
- The decision is **binary**: a word is deleted or kept, with nothing in between

> **IDF REPLACES THE LIST**
>
> A weight of 0.0020 is not deletion, it is near-deletion — and it is *computed from this corpus*. If a word is common here, it is discounted here. Nobody maintains anything.

This is the same move as learning a representation instead of engineering features, arriving thirty years earlier.

Length normalisation, and the parameter b

$$\mathrm{tf}' = \frac{\mathrm{tf}}{\mathrm{tf} + k_1 \left(1 - b + b \, \frac{|d|}{\mathrm{avgdl}}\right)}$$

- $b = 0$ — no normalisation; long documents keep their advantage
- $b = 1$ — length divided out completely

b	0.0	0.4	0.75	1.0
NDCG@10	0.6861	0.6937	0.6917	0.7011

The spread is small, and that is the finding: SciFact abstracts are of similar length (225.2 tokens on average), so there is little for this correction to correct. On web pages there would be.

BM25, assembled

$$\mathrm{BM25}(q,d) \;=\; \sum_{t \in q} \mathrm{idf}(t)\; \frac{\mathrm{tf}_{t,d}\,(k_1+1)}{\mathrm{tf}_{t,d} + k_1\left(1 - b + b\,\dfrac{|d|}{\mathrm{avgdl}}\right)}$$

The sum runs over the query's tokens, repeats included: a term the query uses twice contributes twice.

- The **idf** factor: rare terms count for more
- The **saturating tf**: the tenth occurrence adds far less than the second, because k_1 bounds the growth
- The **length term**: b decides how much of the document's length is divided out

✗ Every piece is there because a specific failure demanded it. That is why BM25 has survived thirty years of replacement attempts, and it is the baseline every neural retriever is measured against — including the one in Lecture 20.

BM25, measured

Metric	Fixed order	BM25
Precision@1	0.0000	**0.5433** ✓
Recall@10	0.0033	**0.7784** ✓
Recall@100	0.0117	**0.8852** ✓
MRR	0.0017	**0.6350** ✓
NDCG@10	0.0012	**0.6611** ✓

The right abstract is in the top ten for **77.8%** of claims, and top of the list for **54.3%**. No training, no labels, no gradient — a data structure and two corrections.

Worth holding on to: **recall@100 is 88.5%**. Almost everything relevant is retrieved *somewhere* in the first hundred. That gap between @10 and @100 is exactly what re-ranking exists to close, in the next lecture.

Which correction was doing the work?

Three corrections were argued for. Every model needs an ablation, because an argument is not a measurement. Each removed in turn, full test set:

Scoring function	NDCG@10
raw term frequency	0.0944
no idf	0.5350
no saturation, no length	0.4801
no length normalisation	0.6490
BM25	**0.6611**

Counting raw words scores 0.0944 — barely above returning the corpus in its stored order.

What the ablation actually says

- **idf is most of the method.** Removing it costs 0.1261; without it, common words drown the rare ones that identify the document
- **Saturation is the one we did not isolate.** Dropping saturation and length together costs 0.1809, and length alone costs 0.0120 — so most of that gap is saturation, but an experiment that removes two things at once cannot say how much
- **Length normalisation is nearly free here**, worth 0.0120 — consistent with the b sweep, and with abstracts being of similar length

> **THE TRANSFERABLE PART**
>
> Not the ordering — that is a fact about SciFact. It is that **each term of a formula can be removed and priced**, and that a component nobody has priced is a component nobody has justified. The same procedure applies to every architecture in Part IV.

The failure BM25 cannot fix

"In mice, P. chabaudi parasites are able to proliferate faster early in infection when inoculated at lower numbers than when inoculated at high numbers."

This claim has exactly one relevant abstract in the corpus. BM25 ranks it ✗ **4,999** of 5,183 — the worst placement of any single-evidence claim in the test set.

Query terms in the vocabulary	19
Terms the relevant abstract contains	✗ **2**
Which terms	`in`, `to`
Mean idf of the shared terms	0.0378
Mean idf of the missing terms	3.2928

✗ The abstract shares `in` and `to` with the claim it is evidence for, and those carry an idf of 0.0378. There is nothing for BM25 to match on. No parameter setting rescues this.

The ceiling, and what is left on the table

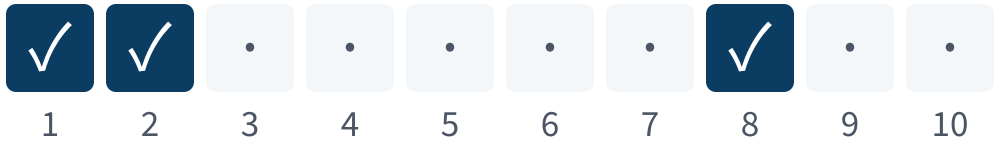
Metric	BM25	The most it could be
Recall@10	0.7784	1.0000
Recall@100	0.8852	1.0000
NDCG@10	0.6611	1.0000

Two different opportunities, and they need different methods. Between recall@10 (77.8%) and recall@100 (88.5%) sit documents BM25 *found* but ranked too low — a **re-ranking** problem, solvable by an expensive model on a hundred candidates.

The remaining 11.5% were never retrieved at all. No re-ranker can reach them; only a first stage that matches on something other than words can. Those are the two halves of Lecture 20.

Precision and recall, at a cutoff

For one query we are given two things: the system's **ranked list** $d_1, d_2, \ldots$ and the set R of **relevant documents**. Throughout, one ranking: $|R| = 3$, and the relevant ones sit at ranks **1, 2 and 8**.



- $\mathrm{P@}k = \dfrac{|R \cap \text{top-}k|}{k}$

  of what I showed you, how much was good
- $\mathrm{R@}k = \dfrac{|R \cap \text{top-}k|}{|R|}$

  of what was good, how much I showed you

With $k = 10$, all three are inside the top ten: $\mathrm{R@10} = \frac{3}{3} = 1$, **perfect recall**. **✗ Yet the third sits at rank 8, where a user may never look. Recall counts a set; it does not read an order.**

Reciprocal rank: where is the first one?

$$\mathrm{RR} = \frac{1}{\text{rank of the first relevant document}}$$

Our example finds one at rank 1, so $\mathrm{RR} = 1/1 = 1.0000$. Averaged over queries this is **MRR**, and over all 300 claims BM25 scores 0.6350.

First hit at rank	1	2	5	10
RR	1.0000	0.5000	0.2000	0.1000

Smooth, bounded, and it moves when the order moves. But it stops reading after the first hit — our ranking would score 1.0000 whether the other two abstracts were at ranks 2 and 8 or nowhere at all.

Which makes MRR right for one kind of question — “is there a good answer near the top?” — and wrong for “did I find the evidence?”

Average precision: read every hit

$$\mathrm{AP} = \frac{1}{|R|} \sum_{i \,:\, d_i \in R} \mathrm{P@}i$$

THE PRECISION AT EACH RANK WHERE A RELEVANT DOCUMENT APPEARS

Hit at rank i	Relevant so far	P@i
1	1	1.0000
2	2	1.0000
8	3	0.3750

$\mathrm{AP} = \frac{1}{3}(1.0000 + 1.0000 + 0.3750) = 0.7917$ — every hit contributes, and one further down contributes less, because the precision above it is worse.

NDCG: discounted gain, normalised

$$\mathrm{DCG@}k = \sum_{i=1}^{k} \frac{g_i}{\log_2(i+1)}$$

A **gain** g_i for how relevant the document at rank i is (here 1 or 0), and a **discount** for how far down it sits. The $+1$ keeps rank 1 from dividing by zero; the logarithm itself is a convention about users, not a derived result.

Hit at rank	1	2	8	DCG@10
contribution	1.0000	0.6309	0.3155	**1.9464**

Divide by the **ideal** DCG — all three relevant at ranks 1–3, IDCG@10 $= 2.1309$ — so queries with different numbers of relevant documents become comparable: $\mathrm{NDCG@10} = 1.9464/2.1309 = \mathbf{0.9134}$.

Same ranking	Value
Recall@10	✗ **1.0000**
RR	1.0000
AP	0.7917
NDCG@10	**0.9134**

✗ **The metric you report decides what you are allowed to notice.**

Which one to report

Question	Metric
Is there a good answer at the top?	MRR
Did I find everything, order aside?	Recall@k
Did I find everything, order counted?	AP
Position counted, relevance possibly graded?	NDCG@k
Will the second stage have anything to work with?	Recall@100

THE RULE

Choose the metric from the **question**, before you see any scores. Choosing it afterwards, from the numbers, is how a paper reports recall@100 for the system that lost on NDCG@10 — and it is the single commonest dishonesty in this literature.

Averaging over queries

One more choice, and it is usually made silently. Given per-query scores, do you average the **scores**, or pool the **counts**?

- **Macro** — mean of the per-query metric. Every query counts equally, including one with a single relevant document
- **Micro** — pool hits and totals across queries first. Queries with many relevant documents dominate

Every number in this lecture is **macro**: the mean over 300 claims of that claim's own metric. MRR and NDCG are essentially always reported that way; precision and recall sometimes are not, and the two can differ substantially.

The variance across queries is large and almost never reported. A mean NDCG of 0.6611 contains claims scoring 1 and claims scoring 0.

What none of these metrics measure

- **Latency.** A ranking that takes a minute has a real score of zero, and no metric here notices
- **Redundancy.** Ten identical relevant documents score perfectly and serve the user once
- **The unjudged.** Anything the pool never saw is counted as irrelevant — the subject of the last section
- **Harm.** A confidently ranked wrong answer and a merely absent one score identically

> **REPORT THE FAILURE CASES, IN THIS SETTING**
>
> Every one of these is a real requirement that the reported number is silent about. A metric is a *choice of what to look at*, and it is therefore also a choice of what to ignore.

What a relevance judgement is

Every metric so far needed the set R. Where does it come from? **A person decided.** A *relevance judgement* is one recorded decision about one pair:

query	document	judgement
a claim	an abstract	1 = relevant, 0 = not

- **Relevant** means: a reader with this query would want to see this document. In SciFact, an abstract is relevant to a claim when it contains evidence about it — *for or against*. Relevant does not mean "agrees"
- The judgements for a test set are stored as a table of such rows, called the **qrels** (query relevance judgements)
- R for a query is simply the documents its qrels mark relevant

> **SO EVERY SCORE IS A COMPARISON**
>
> NDCG, AP, MRR: each compares the system's ranking with somebody's judgements. If the judgements are wrong or incomplete, the score is too — and nothing in the arithmetic will tell you.

Nobody judges every pair

To know R exactly, someone would have to read every abstract against every claim:

300 claims × 5,183 abstracts	1.55 million pairs
Pairs judged relevant in SciFact's test qrels	**339 ✓**
As a fraction of all pairs	0.0218%

That is for a *small* corpus. For a web collection, judging every pair is not expensive; it is impossible. So in practice only a small set of pairs is ever looked at, and a convention covers the rest:

> **THE CONVENTION UNDER EVERY NUMBER IN THIS LECTURE**
>
> A pair that nobody judged is scored as **not relevant**. Not because anyone read it and said no — because nobody read it at all.

So the real question is *which* pairs get looked at. The next two slides show the standard answer, pooling, and the one SciFact used.

Pooling, and what it misses

The standard way large benchmarks choose what to judge:

1 Run
several existing systems on every query

2 Pool
take each one's top k, and merge

3 Judge
people label only the pooled pairs

- A relevant document that **no pooled system** ranked highly is never judged — so it is scored as irrelevant
- A **new method** that finds it is then charged with a miss: it is **penalised for succeeding**
- The pool was built from the systems of its day, so an old benchmark quietly favours methods that resemble them

FAILURE CONDITION — EVERYTHING UNJUDGED COUNTED AS IRRELEVANT

The metric cannot tell "retrieved an unjudged relevant document" from "retrieved junk". Gains on a mature benchmark are partly gains at *matching the pool*, which is why a leaderboard difference of a point or two is rarely worth defending.

How SciFact's judgements were made

SciFact did not pool. Its judgements come from **citations**:

1. Start from a sentence in a published paper that cites other papers
2. An expert annotator rewrites that sentence as a short, checkable **claim**
3. Annotators read the abstracts **that sentence cited**, and mark each one that supports or refutes the claim
4. Those marked abstracts become the claim's relevant documents

RELIABLE

Experts read each cited abstract and marked the sentences carrying the evidence: a judgement records evidence, not a hunch from a title.

SPARSE

Only the cited abstracts were ever read: 1.13 relevant per claim, and 92.3% of claims have exactly one. An uncited abstract with the same evidence counts as a miss.

✗ So read BM25's 0.6611 as *agreement with what the claims' sources cited*, not as how useful its ranking is to a reader. An uncited abstract carrying the same evidence, ranked first, is scored exactly like junk.

Graded relevance: when relevant has degrees

SciFact's judgements are **binary**: 1 or 0. Many collections instead ask judges for a **grade**, because an abstract that directly tests a claim and one that mentions it in passing are not equally useful.

The grade becomes the gain g_i in DCG. Two common choices:

Grade rel_i	Linear gain rel_i	Exponential gain $2^{\mathrm{rel}_i} - 1$
0 — not relevant	0	0
1 — fair	1	1
2 — good	2	3
3 — excellent	3	7

With binary judgements the two coincide, which is why our NDCG never had to choose. With grades they do not: the exponential makes one excellent document worth seven fair ones, the linear only three.

Graded relevance changes the answer

One query, two systems, top three results. Discounts $1/\log_2(i+1)$ are $1, 0.63$ and 0.5 at ranks 1, 2 and 3.

	Rank 1	Rank 2	Rank 3	DCG, linear	DCG, exponential
System A	fair	fair	not relevant	**1 + 0.63 = 1.63 ✓**	1 + 0.63 = 1.63
System B	not relevant	not relevant	excellent	3 × 0.5 = 1.5	**7 × 0.5 = 3.5 ✓**

Same query, so the same ideal DCG divides both, and the order of the NDCGs is the order of the DCGs. **Linear gain prefers A; exponential gain prefers B.** Nothing about the systems changed — only the gain function.

> **WHAT TO DO WITH THAT**
>
> The exponential says one excellent result at rank 3 beats two fair ones at the top. That is a claim about users, not about mathematics. Whoever reports NDCG on graded judgements must say which gain they used, and be ready to defend it.

The notebook for this lecture

Open Lecture 19 in Colab

- **Runs on free CPU.** No model is trained — there is nothing here to train
- The inverted index built by hand from a dictionary, and BM25 scored exactly against every abstract
- All the metrics implemented from the derivation, checked by hand against the ranking used on these slides, and the qrels counted against the full grid of pairs

> **WHAT TO CHANGE**
>
> Add the ablation’s missing row: saturation removed, length normalisation kept. Beside the no-length row, it says how much of the no-saturation gap is saturation alone.

Design exercise · in the manner of an ML system-design interview

The task: a legal-tech company’s search product for law firms

> **THE BRIEF**
>
> A legal-tech company sells search to law firms. Its product indexes all a firm has written — memos, contracts, briefs, opinions — and for a lawyer’s query returns a **ranked list of documents**, of which they read the top few before the client calls. Firms pay per seat and renew if lawyers find things; a missed precedent is a client’s loss, and a document shown across an ethical wall ends the contract.

- **Given:** each firm’s document store, decades deep, with each document’s matter and who may see it; the old system’s query log
- **Scale:** hundreds of firms; millions of documents each, letters to long contracts; thousands of queries a day per firm
- **Consumers of the output:** the lawyers, each firm’s knowledge team, and later a second stage that re-ranks its output

✓ Nine steps follow, each with the pitfall that catches a first attempt and what to do about it. The model at the centre is today’s: an inverted index and BM25, scored against judgements somebody has to make.

Step 1 — Frame it: the output is an ordering, and the question picks the metric

What the lawyer is doing	The question	Metric
Finding one precedent to start from	is there a good answer at the top?	MRR
Due diligence on a matter	did I find everything, order aside?	recall at a deep cutoff
Reading a results page	position counted, relevance graded?	NDCG at ten

PITFALL

Choosing the metric after seeing the scores — sales quoting the deep recall for the system that lost on NDCG. Two uses, two questions, two metrics, written down with the pilot firm before a query is run, each with its cutoff: MRR rewards one good answer at the top and says nothing of the rest; NDCG reads the page and discounts by position. And the comparison is with the old system on the same queries, since the fixed corpus order scores near zero.

Step 2 — The corpus: what a document is, and how many times it exists

- A document is the text the index can see — a scanned letter with no text layer is invisible, and so is an email attachment never extracted. Count what is indexable first, and tell the firm the number
- The same memo saved in five matters; a contract in a dozen drafts and one signed version; an opinion superseded by a later one that cites it

PITFALL — STALE AND DUPLICATED DOCUMENTS

The top ten fills with the same contract in nine drafts and the signed version just below the fold: the metric scores nine relevant hits and the lawyer is served once. Worse, a superseded opinion outranks the one that replaced it, and the lawyer who relies on it is the firm's liability. Collapse duplicates at indexing by normalised text, keep one entry per version chain with the current version on top, and record "superseded" as a field the ranking can read.

Step 3 — Relevance judgements: who decides, on which pairs

- A judgement is one recorded decision about one query–document pair, made by someone who could use the document: a lawyer in that practice area at the pilot firm, not the vendor's engineers
- Graded, because a precedent directly on point and a letter that mentions the issue are not equally useful — the gain per grade written down with the metric
- Nobody judges every pair: pool the top results of several systems for each query, judge the pool; everything outside it counts as not relevant

PITFALL

The pool built from the old system alone. A document only the new product finds was never judged, so it is scored as junk: the new product is penalised for succeeding, and the benchmark quietly prefers whatever resembles the incumbent. Pool the old system, the new one and a plain keyword search; judge to a stated depth; and when the new system puts an unjudged document at the top, have it judged before the number is believed.

Step 4 — The queries: from the log, and split before anything is tuned

- The log says what lawyers actually type — party names, clause titles, two-word fragments — so judged queries are sampled from it, head and tail
- Split the judged queries: a development set on which the tokeniser decisions and the parameters are chosen, and a held-out set read once
- A click is a weak signal, not a judgement: the first result is clicked because it is first; a document the old system never returned was never clicked

> **PITFALL**
>
> Using the clicks as the relevance judgements. Attention falls off the page, so clicks reward what the old system ranked first; the evaluation then measures agreement with the incumbent, and the ranking the firm is paying for loses. Use the log to choose which queries to judge and their weights, and let a lawyer’s reading — not a click — be the judgement.

Step 5 — The method: terms, an inverted index, and BM25 measured here

- Decide what a term is by looking at the text: lower-casing, hyphenated terms, defined terms in capitals, clause references and case citations — the tokeniser decides what can be found
- One postings list per term; idf derives the stop words from this corpus — here *agreement*, *party* and *shall*, which no shipped list contains
- Saturation bounds what repetition can buy; length normalisation matters here as it did not on abstracts, because a letter and a contract differ in length by orders of magnitude

PITFALL

The two parameters inherited from a textbook, or from the last customer's corpus, on which neither mattered. On documents of such uneven length the length parameter decides whether every long contract outranks every short letter. Sweep both on the firm's development queries, report the corpus with the parameter, and keep the sweep as the record of what each is worth.

Step 6 — The failure BM25 cannot fix: the words are different

- The query says *non-compete*; the precedent says *restrictive covenant*. A conjunction returns an empty page; BM25 returns the documents that share *the* and *of*
- Recall at a deep cutoff is the ceiling: what the first stage never returns, no later stage can recover

> **PITFALL**
>
> Reading the useless page as “no such document exists” — which is what the lawyer concludes, what renewal turns on, and what a mean NDCG hides: the failing queries score zero inside an average that looks fine. Report deep recall beside the top-ten metrics, list the queries whose relevant documents share no informative term, and hand that list to a first stage that matches on something other than words — the next lecture. Until then, the firm’s synonym pairs added to the query are the patch.

Step 7 — Evaluate, and price every component

- Macro over the held-out queries, each metric with its cutoff, against the old system on the same queries; the spread across queries reported, since the mean hides the zeros
- An ablation: each of the three corrections removed in turn and the change in NDCG written down, so each is justified by a measurement, not an argument
- Latency per query measured beside the metrics: a ranking that takes a minute has a real score of zero

> **PITFALL**
>
> A component added by argument. A customer asks for a boost for recent documents, sales for a bonus on title terms, an engineer for a phrase match — each sounds right, each ships, and the formula grows a term nobody has priced. A component nobody has priced is a component nobody has justified: every addition gets its ablation row on the development queries, and stays only if the row says so.

Step 8 — In production: who may see what, decided before the ranking

- Every document carries who may see it — matters walled off from lawyers on the other side of a conflict — and that changes as matters open and close
- The filter runs inside retrieval, as the postings are read, so the top ten is the top ten *of what this lawyer may see*
- Roll out firm by firm, old beside new on the pilot's queries; re-index on a schedule, track staleness and empty results; an engineer on call

> **PITFALL**
>
> Ranking first, filtering afterwards. The page arrives with gaps where forbidden documents were, or — worse — their titles, a breach of the wall by the search box, reported to the regulator; and every metric was computed on a ranking nobody sees. Filter at retrieval, evaluate as each judging lawyer sees it, and test the wall in the release suite with a query that must return nothing.

Step 9 — Ship it: the pipeline, and the pitfalls in one table

Document store → text extracted, duplicates collapsed, versions chained → log queries, judged by lawyers on a pool → development and held-out queries → tokeniser, inverted index with the access list → BM25, parameters swept per firm → MRR, NDCG, deep recall, an ablation, a latency column → filtered at retrieval → rollout per firm, re-index, monitor, mismatch to the next stage.

Pitfall	Address it by
The metric chosen from the numbers	the question first; MRR, NDCG, deep recall each for its use
Stale and duplicated documents	collapsed by text; version chains; a superseded field
A pool from the old system only	several systems pooled; unjudged hits at the top judged
Clicks as judgements	the log to choose queries; a lawyer’s reading as the judgement
Parameters inherited from another corpus	both swept on the development queries
Vocabulary mismatch read as absence	deep recall reported; failing queries listed
A component nobody priced	an ablation row for every addition

Summary

1. Retrieval returns an **ordering of a corpus**, not a label; the metrics, baselines and methods all follow from that
2. The inverted index makes it cheap: postings are 0.3% of the dense matrix
3. BM25 is three corrections to counting words, each answering a stated failure, and it scores **0.6611** NDCG@10 with no training at all
4. A ranking metric answers a **question**. Recall@10 called our example perfect while a relevant document sat at rank 8
5. Every one of those numbers is a comparison with relevance judgements: decisions people made on a small set of pairs, with everything unjudged counted as irrelevant
6. Designed a legal-tech company's search product for law firms in nine steps — the question before the metric, duplicates and superseded versions collapsed, judgements made by the pilot firm's lawyers on a pool of several systems, the query log used to choose queries and not as judgements, both BM25 parameters swept per firm, mismatch reported as deep recall, every component priced, and access control inside retrieval, rolled out firm by firm

NOT PRESENTED — FOR AFTERWARDS

Exercises Lecture 19

five, in the style of the written paper

Exercises — Lecture 19

1 A ranking has relevant documents at ranks 1, 3 and 10, with three relevant documents in total. Compute AP and NDCG@10, showing the sums. [6]

2 State why $\log_2(i+1)$ is used rather than $\log_2(i)$ in DCG, and whether the choice of discount is derived or conventional. [4]

3 BM25 has three parts. Name the failure of raw term counting each one answers, then say which part the ablation prices on its own as costliest, and which part it cannot price alone. [5]

Solutions on Lecture 20’s deck.

Exercises — Lecture 19 (continued)

4 98% of claims match no document under Boolean conjunction. State the property of the conjunction that causes it, and what the field does instead. [4]

5 Large benchmarks choose which pairs to judge by pooling. State what happens to a document no pooled system ranked highly, and the consequence for a genuinely new method. [4]

Solutions on Lecture 20’s deck.

THE FIVE SET AT THE END OF LECTURE 18

Solutions Lecture 18

not part of this lecture

Solutions — Lecture 18

1. Scaled dot-product attention divides by $\sqrt{d_k}$. State what the variance of the unscaled dot product is,…

✓ $\mathrm{Var}(s) = d_k$, so s has standard deviation $\sqrt{d_k}$; without the division the softmax saturates and the gradient vanishes.

- A sum of d_k independent zero-mean products, each of variance 1, has variance d_k — at $d_k = 64$ typical scores are already of order 8
- Large logits make the distribution nearly one-hot, so almost no gradient flows

2. A decoder must not attend to positions after the one it is predicting. State the mechanism that enforces it,…

✓ A mask applied to the attention scores before the softmax, setting future positions to $-\infty$. Without it the model sees the token it is being asked to predict.

- Attention is all-to-all by construction, so nothing in the mechanism itself respects order
- The training loss would fall and the model would be useless at generation time, when the future genuinely does not exist

Solutions — Lecture 18 (2)

3. A pretrained body is fine-tuned at 10^{-3}, the rate that worked for the from-scratch model. Predict what…

✓ The first steps overwrite the pretrained features, and the model is left to relearn them from the labels alone, badly. The lecture fine-tunes at 2×10^{-5}.

- A pretrained block already encodes something; a step tuned for random starting weights is far too large for weights that are already nearly right
- Fifty times smaller than the 10^{-3} that trained the model from scratch — large steps destroy what was pretrained

4. A tf-idf vectoriser is fitted on every document of a 400-review corpus, and the matrix is split afterwards.…

✓ On the training rows only, inside the pipeline. A null result, and a leaked score can equal an honest one, so the number cannot tell you which you are holding — only the procedure can.

- 0.30 points on 400 reviews is not distinguishable from zero: the seeds fall on both sides of it
- A small number is not a clean one: the procedure is what separates a leaked score from an honest one, which is why the procedure is what gets checked in review

Solutions — Lecture 18 (3)

5. A corpus contains duplicate documents across the train/test split. State why no pipeline fixes it, and say…

✓ Nothing was fitted wrongly — the rows were separated wrongly. Split by entry, so that every copy of a document falls on the same side.

- `fit` and `transform` were used correctly throughout
- The difference is entirely which rows the split separates: measured, the random split reads 7.83 points higher than the split grouped by entry

LECTURE 20 · LECTURE NOTES

Information retrieval: dense retrieval

Bi-encoders: one vector per claim and per abstract, and how they are trained
Measured against today’s BM25, on the same judgements — and where the two disagree
Hybrid retrieval: fusing the two rankings
Cross-encoders and re-ranking — why the accurate model cannot go first

Part V · Lecture notes · examinable