Lecture 19 · Part V · Extended notes

Information retrieval: the lexical foundation

Dataset: SciFact (BEIR), 5,183 abstracts

Why these notes exist

Lectures 19–22 sit outside Géron, the course's primary text. For those four lectures *these notes are the primary source*, and they are examined on exactly the same terms as the textbook parts. Everything in the slide deck for this lecture is here, with the arguments written out at length rather than compressed onto a slide; the numbers are the ones the accompanying notebook prints, and every one of them is a measurement on the corpus named above rather than a figure quoted from a paper.

Contents

Part V begins **2**

1 The problem **3**

1.1 The shape of every search system 3

1.2 An assumption we import 4

1.3 The corpus, counted 4

1.4 The baseline: return the corpus unranked 4

2 From filtering to scoring **5**

2.1 First: what is a term? 5

2.2 The obvious first idea, and its measurement 5

2.3 Why retrieval is cheap 5

3 Term weighting, and BM25 **6**

3.1 Saturation, and the parameter k_1 6

3.2 Inverse document frequency 7

3.3 Length normalisation, and the parameter b . . . 7
3.4 BM25, assembled . . . 8
3.5 BM25, measured . . . 8
3.6 Which correction was doing the work? . . . 8

4 The failure BM25 cannot fix **9**
4.1 The ceiling, and what is left on the table . . . 9

5 Derivation: evaluating a ranking **10**
5.1 Precision and recall, at a cutoff . . . 10
5.2 Reciprocal rank . . . 10
5.3 Average precision . . . 10
5.4 Discounted cumulative gain, and its normalisation . . . 11
5.5 Which one to report . . . 12
5.6 Averaging over queries . . . 12
5.7 What none of these metrics measure . . . 13

6 Where the judgements come from **13**
6.1 What a relevance judgement is . . . 13
6.2 Nobody judges every pair . . . 13
6.3 Pooling, and what it misses . . . 14
6.4 How SciFact's judgements were made . . . 14
6.5 Graded relevance . . . 14

7 The notebook **15**

8 Design exercise: a legal-tech company's search product for law firms **15**

9 Exercises **17**

10 Summary **18**

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

Four lectures the textbook does not cover, because search and recommendation are the two applications of machine learning most of you will actually meet, and because they are where the embeddings of Part IV earn their living. They are outside the book and examinable, on exactly the same terms as everything else; these notes are their primary source.

What you should be able to do afterwards.

- State why retrieval is not classification, in terms of what the output is.
- Build an inverted index, and say what it costs and what it saves.
- Write BM25 down, and say what each of its three parts corrects for.
- Compute P@k, R@k, RR, AP and NDCG@k by hand from a relevance pattern.
- Choose between them from the question being asked, and defend the choice.
- Say what a reported retrieval score does not tell you.

1 The problem

Eighteen lectures of this course have been about prediction: given an input, produce a label, a number, or a sequence. Retrieval is not that.

The input is a **query** and a **corpus**. The output is an **ordering** of the corpus. Nobody wants a yes or a no; they want the evidence, ranked, so that a person can read the top few and decide.

That change of output changes everything downstream of it. There is no fixed set of classes, because the corpus changes daily. There is no single right answer, because several documents may be relevant and their order matters. And you cannot afford to score the whole corpus with a neural network for every query, because there are millions of documents and the user is waiting.

1.1 The shape of every search system

Every production search system has the same three stages.

1. **Retrieve**: millions → hundreds. Must be cheap.
2. **Re-rank**: hundreds → tens. Can be expensive.
3. **Present**: tens → ten. Diversity, freshness, policy.

This lecture is stage 1, and stage 1 decides the ceiling:

The first-stage ceiling

A document the first stage never returns *cannot* be recovered by any later stage, however good that stage is. This is why we will keep quoting 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.

Stage 2 is Lecture 20. Stage 3 is mostly not machine learning and mostly not in this course.

1.2 An assumption we import

Whatever we build, a person reads its output, and that person's attention is not uniform: the first result is examined far more often than the second, almost nobody reaches the tenth, and the second page effectively does not exist.

Nothing in this lecture *measures* that. It is an assumption imported from outside, and it is flagged as one here because every metric in Section 5 is an attempt to encode it — and the differences between those metrics are differences of opinion about the shape of the decay.

1.3 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%

The last row matters more than it looks. With one relevant document per claim most of the time, recall@10 and the reciprocal rank measure nearly the same thing. On a corpus where every query had twenty relevant documents, they would not, and half of Section 5 would read differently.

SciFact is used here for one reason: it 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 rather than of an index setting nobody inspected.

A document is a title and an abstract and nothing else — no full text, no citations, no authors; one typical abstract runs to 311 tokens. Notice how they are 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 returns, with a number attached, in Section 4.

1.4 The baseline: return the corpus unranked

Every number needs a baseline, here as anywhere. Before any method, ask what *no* method scores. Return every document in its stored corpus order, the same order for every query:

Metric	Fixed order
Recall@10	0.0033
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. So "better than nothing" is never an argument in retrieval: the comparison that matters is always against the *previous method*.

2 From filtering to scoring

2.1 First: what is a term?

Before any weighting there is a decision almost nobody writes up: how the text becomes a list of things to count. Take a real claim from the corpus, "0-dimensional biomaterials show inductive properties." Lower-case it and split on non-alphanumeric characters and you get six tokens:

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

That is already three decisions, and one of them has done damage: `0-dimensional` has become `0` and `dimensional`, two terms that mean nothing apart. No weighting scheme has been chosen yet, and the tokeniser has already decided what can be found.

2.2 The obvious first idea, and its measurement

Boolean retrieval returns the documents containing *all* the query terms:

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

It is exact, it is fast, and it is how search worked for thirty years. Before reading on, predict: on a corpus of scientific abstracts, and claims written by scientists, how many abstracts contain every word of a claim?

Failure condition — Boolean retrieval, over all 300 test claims

98.0% of claims return an *empty page*. Not a bad ranking — nothing at all. And the failure is silent, because an empty result set is indistinguishable from "no such document exists".

The conjunction is brittle: one word of the claim absent from an otherwise perfect abstract, and the abstract is excluded entirely. This is the observation that reorganises the whole field:

Stop filtering; start scoring

Every document gets a number. The user gets the largest ones. A missing term then costs *something* rather than *everything*.

2.3 Why retrieval is cheap

Scoring every document for every query by reading each one is hopeless at scale. The trick that makes it cheap is fifty years old, and it is a data structure rather than a model.

Definition 2.1 (Inverted index)**.** *For every term, store the list of documents containing it, together with the count. One term's entry is called its* ***postings list****.*

A query of four words then touches only the documents containing at least one of those four words, never the rest of the corpus: a document sharing no term with the query contributes exactly zero to any sensible score, so it need not be read.

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

The postings occupy 0.3% of the dense matrix. Sparsity here is not an optimisation; it is the reason the field is possible at all. What remains to be decided is how much each matching term should add to a document's score.

3 Term weighting, and BM25

Start from the obvious idea: 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 arrive immediately.

1. **Common words dominate.** Every abstract contains *of* and *the*, so a query containing them matches everything equally.
2. **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 rather than argued about.

3.1 Saturation, and the parameter k_1

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

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

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

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

k_1 sets how fast the saturation bites: small k_1 saturates almost immediately, large k_1 approaches raw counting. Measured over 150 claims with 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. That is the honest finding, and it is not the one you would guess from how much has been written about tuning this parameter: on abstracts, k_1 barely matters. It matters on long documents, where a term can occur fifty times. *Report the corpus with the parameter, always.*

3.2 Inverse document frequency

The second correction weights a term by how rare it is:

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

for N documents of which n_t contain t.

Term	Abstracts	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.

idf replaces the stop-word list

Classical systems shipped a hand-written list of words to delete. That list is language-specific and must be maintained; it is corpus-blind, so in a corpus about the band *The Who* deleting *the* and *who* destroys the query; and its decision is binary, with nothing between delete and keep.

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, and nobody maintains anything. This is the same move as learning a representation instead of engineering features — arriving thirty years earlier.

The particular form above is used rather than the older $\log(N/n_t)$ because it stays well-behaved for a term appearing in almost every document. Whichever you use, say which: the two agree closely where n_t is small and diverge exactly where a stop word lives.

3.3 Length normalisation, and the parameter b

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

with $b = 0$ meaning no normalisation, so long documents keep their advantage, and $b = 1$ meaning length is 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 again small, and again that *is* the finding: SciFact abstracts are of similar length — 225.2 tokens on average — so there is little here for this correction to correct. On web pages, whose lengths span three orders of magnitude, there would be.

3.4 BM25, assembled

BM25

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

The sum runs over the query's tokens, repeats included: a term the query uses twice contributes twice. This is the convention of the notebook, and every score below uses it.

- 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.

3.5 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.

Hold on to recall@100: 88.5%. Almost everything relevant is retrieved *somewhere* in the first hundred. The gap between @10 and @100 is exactly what re-ranking exists to close, and it is half of Lecture 20.

3.6 Which correction was doing the work?

Three corrections were argued for. Every model needs an ablation, because an argument is not a measurement. Remove each correction 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

- **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, which is 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.

4 The failure BM25 cannot fix

The worst single-evidence claim in the test set

"In mice, P. chabaudi parasites are able to proliferate faster early in infection when inoculated at lower numbers than when inoculated at high numbers."
BM25 ranks its one relevant abstract **4,999 of 5,183**.

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. There is nothing for BM25 to match on, and no parameter setting rescues this.

4.1 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

There are two different opportunities here 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 over a hundred candidates. The remaining 11.5% were never retrieved at all; no re-ranker can reach them, and only a first stage that matches on something other than words can. Those are the two halves of Lecture 20.

5 Derivation: evaluating a ranking

Every number so far — recall@10, MRR, NDCG@10 — has been quoted without saying how it was computed. This section derives them.

For one query we are given two things: the system's **ranked list** $d_1, d_2, \ldots, d_n$, an ordering of the corpus, and the set R of **relevant documents**, taken from the judgements. Everything below is a function of those two objects and nothing else — no scores, no probabilities, no model — which is why the same metrics apply to any retriever. Section 6 is about where R comes from, and what it costs to treat it as complete.

Throughout, one ranking: three relevant documents, $|R| = 3$, placed at ranks 1, 2 and 8.

✓	✓	·	·	·	·	·	✓	·	·
1	2	3	4	5	6	7	8	9	10

5.1 Precision and recall, at a cutoff

$$\mathrm{P@}k = \frac{|R \cap \text{top-}k|}{k}, \qquad \mathrm{R@}k = \frac{|R \cap \text{top-}k|}{|R|}.$$

P@k asks: of what I showed you, how much was good. R@k asks: of what was good, how much did I show you.

On the example with $k = 10$, all three relevant documents are inside the top ten, so R@10 = $3/3 = 1$: perfect recall. And that is the problem. The third sits at rank 8, where a user may never look, and recall@10 cannot say so. It counts a *set*; it does not read an *order*.

5.2 Reciprocal rank

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

Our example finds one at rank 1, so RR = 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 scores 1.0000 whether the other two abstracts sit at ranks 2 and 8 or nowhere at all. That makes MRR right for "is there a good answer near the top?" and wrong for "did I find the evidence?".

5.3 Average precision

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

— 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

$$\text{AP} = \tfrac{1}{3}\,(1.0000 + 1.0000 + 0.3750) = 0.7917.$$

Every hit contributes, and one further down contributes less because the precision above it is worse.

Failure condition — where marks are lost

Divide by $|R|$, not by the number of hits actually found. Dividing by the hit count gives *mean precision*, which is a different and more flattering quantity, and it is the commonest error in this material.

5.4 Discounted cumulative gain, and its normalisation

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

Two independent parts: 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, since $\log_2 1 = 0$. The logarithm itself is not derived from anything: it is a convention, a belief about how fast a reader's attention falls down the page. If you report NDCG, you have adopted that curve.

Hit at rank i	$\log_2(i+1)$	contribution
1	1.0000	1.0000
2	1.5850	0.6309
8	3.1699	0.3155
DCG@10		**1.9464**

Is 1.9464 good? The question has no answer, because DCG depends on how many relevant documents exist: a claim with ten relevant abstracts can reach a far larger DCG than one with three, however well both are ranked. So divide by the best this query could possibly have scored — the ideal ranking, all relevant documents first.

Ideal rank	contribution
1	1.0000
2	0.6309
3	0.5000
IDCG@10	**2.1309**

NDCG

$$\text{NDCG@}k = \frac{\text{DCG@}k}{\text{IDCG@}k} = \frac{1.9464}{2.1309} = 0.9134.$$

The same ranking, under four metrics:

Metric	Value
Recall@10	1.0000
RR	1.0000
AP	0.7917
NDCG@10	0.9134

Recall calls this ranking perfect. It is not: one abstract sits at rank 8. *The metric you report decides what you are allowed to notice.*

5.5 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
Graded relevance, position counted?	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 comes to report recall@100 for the system that lost on NDCG@10 — and it is the single commonest dishonesty in this literature.

5.6 Averaging over queries

One more choice, 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 these notes 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.

5.7 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 — Section 6.
- **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.

Check

Take the relevance pattern ✓ · ✓ · · · · · ✓ (relevant at ranks 1, 3 and 10, three relevant documents in total) and compute P@10, R@10, RR, AP and NDCG@10 by hand. Then check each against `rank_metrics` in the notebook.

6 Where the judgements come from

6.1 What a relevance judgement is

Every metric above needed the set R. Somebody decided what is in it. A **relevance judgement** is one recorded decision about one pair: for this query, this document is relevant (1) or not (0). *Relevant* means that a reader with this query would want to see the document. In SciFact, an abstract is relevant to a claim when it contains evidence about the claim — for it *or against it*. 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), and R for a query is simply the set of documents its qrels mark relevant. Every score in this lecture — NDCG, AP, MRR — is therefore a *comparison* between a system's ranking and somebody's judgements. If the judgements are wrong or incomplete, so is the score, and nothing in the arithmetic will say so.

6.2 Nobody judges every pair

To know R exactly, somebody would have to read every abstract against every claim: 300 claims against 5,183 abstracts is 1.55 million pairs. SciFact's test qrels mark 339 of them relevant, 0.0218% of the grid. That is for a small corpus; for a web collection, judging every pair is not expensive, it is impossible. So 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, but because nobody read it at all.

The real question is therefore *which* pairs get looked at.

6.3 Pooling, and what it misses

The standard answer, used by most large benchmarks, is **pooling**: run several existing systems on every query, take the top k of each, merge them into a pool, and have people judge only the pooled pairs.

Failure condition — everything unjudged counted as irrelevant

- A relevant document that no pooled system ranked highly is never judged, and is therefore scored as irrelevant.
- A genuinely new method that finds it is 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.

The metric cannot tell "retrieved an unjudged relevant document" from "retrieved junk". Reported 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.

6.4 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, together with the sentences carrying the evidence;
4. those marked abstracts become the claim's relevant documents.

Two properties follow. The judgements are *reliable*: experts read each candidate and recorded the evidence, not a hunch from a title. And they are *sparse*: only the cited abstracts were ever read, so there are 1.13 relevant abstracts per claim on average, and 92.3% of claims have exactly one. An uncited abstract carrying the same evidence is unjudged, and scored as a miss.

So BM25's 0.6611 is best read 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. Reporting "BM25 achieves 0.6611" without that sentence is the kind of claim this course exists to stop.

6.5 Graded relevance

SciFact's judgements are binary. 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, and two choices are common:

Grade rel_i	Linear gain rel_i	Exponential gain $2^{\text{rel}_i} - 1$
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, and the choice can reverse a comparison. Take one query and two systems' top three results; the 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	$3 \times 0.5 = 1.5$	$7 \times 0.5 = 3.5$

Both systems answer the same query, so the same ideal DCG divides both and the NDCGs are in the same order as the DCGs. Linear gain prefers A; exponential gain prefers B. Nothing about the systems changed — only the gain function. The exponential says that one excellent result at rank 3 beats two fair ones at the top: 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.

7 The notebook

The notebook runs on free CPU and trains nothing — there is nothing here to train. It builds the inverted index by hand from a dictionary, scores BM25 exactly against every abstract, implements all the metrics from the derivation, checked by hand against the ranking used above, and counts the qrels against the full grid of claim–abstract pairs.

What to change

Add the row the ablation is missing: saturation removed, length normalisation kept — score $\text{idf}(t) \cdot \text{tf} \,/\, (1 - b + b\,|d|/\text{avgdl})$. Set beside the no-length row, it says how much of the no-saturation gap is saturation alone.

8 Design exercise: a legal-tech company's search product for law firms

The lecture closes with a design exercise in the manner of an ML system-design interview, framed as the ML system behind a company's product.

The task. A legal-tech company sells search to law firms. Its product indexes everything a firm has written — memos, contracts, briefs, opinions, correspondence — and for a lawyer's query, a few words or a sentence, returns a ranked list of documents, of which they will read the top few, before the client calls back. Firms pay per seat and renew on whether their lawyers find things; a missed precedent is a lawyer's hours or a client's loss, and a document shown across an ethical wall is a breach the firm answers for and the end of the vendor's contract. Given: each firm's document store, decades deep, with each document's matter and the list of who may see it, and the old system's query log. Hundreds of firms; millions of documents each, from one-page letters to contracts of hundreds of pages; thousands of queries a day per firm. The output serves the lawyers, each firm's knowledge-management team, and later a second stage that re-ranks what the first returns.

1. **Frame it.** The output is an ordering, and the question picks the metric: one precedent to start from asks whether there is a good answer at the top, which is MRR; due diligence asks whether everything was found, order aside, which is recall at a deep cutoff; a results page asks for position counted and relevance graded, which is NDCG at ten. The fixed corpus order scores near zero, so the comparison that matters is with the old system on the same queries. *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.

2. **The corpus.** A document is the text the index can see: a scanned letter with no text layer is invisible, and so is an attachment never extracted, so count what is indexable first and tell the firm the number. The same memo is saved in five matters, a contract exists in a dozen drafts and one signed version, an opinion is superseded by a later one. *Pitfall, stale and duplicated documents:* the top ten fills with nine drafts of one contract and the metric scores nine hits while the lawyer is served once; 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 time 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.

3. **Judgements.** 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, with the gain for each grade written down beside the metric; and pooled, because nobody judges every pair and everything outside the pool counts as not relevant. *Pitfall:* the pool built from the old system alone, so a document only the new product finds is scored as junk and the new product is penalised for succeeding. Pool several systems, judge to a stated depth, and have an unjudged document the new system puts at the top judged before the number is believed.

4. **The query log.** It says what lawyers actually type, so the judged queries are sampled from it, head and tail, and split into a development set on which the tokeniser decisions and the parameters are chosen and a held-out set read once. *Pitfall:* the clicks used as the judgements. Attention falls off the page, so clicks reward whatever the old system ranked first, and the evaluation measures agreement with the incumbent: the ranking the firm is paying for loses. Use the log to choose which queries to judge and how to weight them, and let a lawyer's reading be the judgement.

5. **Index and BM25.** Decide what a term is by looking at the text — hyphenated terms, defined terms in capitals, clause references, case citations. One postings list per term; idf derives the stop words from this corpus, where they are *agreement*, *party* and *shall*. Saturation bounds what repetition can buy, and 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 each firm's development queries and report the corpus with the parameter.

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

can recover. *Pitfall:* the useless page read as "no such document exists", which is what the lawyer concludes, what the renewal turns on, and what a mean NDCG hides. Report deep recall beside the top-ten metrics, list the queries whose relevant documents share no informative term with them, and hand that list to a first stage that matches on something other than words; until then, the firm's own synonym pairs added to the query are the honest patch.

7. **Price it.** Macro over the held-out queries, each metric with its cutoff, against the old system on the same queries, with the spread across queries reported; an ablation that removes each correction in turn and writes down the change; latency per query beside the metrics, since 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 of which sounds right and ships. A component nobody has priced is a component nobody has justified: every addition gets its ablation row, and stays only if the row says so.

8. **In production.** Every document carries the list of who may see it, and the list 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, the old system beside the new on the pilot firm's queries; re-index on a schedule, track how stale the index is and which queries return nothing, and keep an engineer on call. *Pitfall:* ranking first and filtering afterwards. The page arrives with gaps, or with the titles of forbidden documents, which is a breach of the wall by the search box that the firm reports to its regulator, and every metric was computed on a ranking nobody sees. Filter at retrieval time, evaluate as each judging lawyer would see it, and test the wall in the release suite with a query that must return nothing.

9. **Ship it.** Document store → text extracted, duplicates collapsed, versions chained → queries sampled from the log, judged by the pilot firm's lawyers on a pool of several systems, graded → development and held-out queries → tokeniser decided on the text, inverted index with the access list → BM25 with both parameters swept per firm → MRR, NDCG and deep recall, macro, with an ablation and a latency column → filtered at retrieval → rollout firm by firm, re-index, monitor, hand the mismatch queries to the next stage.

9 Exercises

The five set at the end of this lecture's deck, in the style of the written paper. The solutions are on Lecture 20's deck and in the exercise book.

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]

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]

10 Summary

Retrieval returns an ordering of a corpus, not a label, and the metrics, baselines and methods all follow from that. The inverted index makes it cheap: postings are 0.3% of the dense matrix. BM25 is three corrections to counting words, each answering a stated failure, and it reaches 0.6611 NDCG@10 with no training at all. A ranking metric answers a question — recall@10 called our example perfect while a relevant document sat at rank 8. And 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. The closing design exercise built 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.