

What Counting Built

Fifty years of count-based NLP, before we stop counting

James Pustejovsky

Brandeis University
Department of Computer Science

CS 135: Computational Semantics

Fall 2026

Where We Are

Session 7 built word vectors by counting. Session 8 compressed them with the SVD, and then began a sentence that ends badly for counting: *stop counting, start predicting*.

Before we finish that sentence, it is worth asking what the counting tradition actually produced, because the answer is not “a first attempt”.

Today

Four places where counting became working technology: information retrieval, corpus linguistics and lexicography, question answering, and language modeling. Each one contributes a technique we will meet again.

The Claim Worth Resisting

The standard story runs: first people counted words, which was crude; then people learned to predict them, which was not.

That story is wrong in a specific way. Counting was not a preliminary stage that prediction replaced. It was, and in places still is, the substrate of deployed systems.

Built on counts	Still true
Web search ranking	BM25 remains a standard baseline and first-stage retriever
Corpus lexicography	Dictionary entries are still built from concordances
Statistical MT and ASR	<i>n</i> -gram models ran production systems for two decades

The interesting question is not whether counting works. It does. The question is exactly which problem it cannot solve, and that is what the prediction methods were built to attack.

The Retrieval Problem

Given a query and a collection of documents, return the documents in order of usefulness.

Stated that way it is not obviously a counting problem. Salton's move, in the SMART system from the 1960s onward, was to make it one:

The vector space model

Represent each document as a vector of term counts. Represent the query the same way. Rank by the similarity of the two vectors.

You have already built this object. The term-document matrix from Session 7 is exactly an index: rows are words, columns are documents, and the entry is how often the word occurs in the document.

What changes is the question asked of it. Session 7 compared *rows* to find similar words. Retrieval compares *columns* to find relevant documents.

Two Intuitions, Fifteen Years Apart

Luhn, 1958	A word that occurs often in a document is probably what the document is about. Term frequency.
Spärck Jones, 1972	A word that occurs in nearly every document tells you nothing about which one you want. Inverse document frequency.

Put together:

$$\text{tf-idf}(w, d) = \text{tf}(w, d) \times \log \frac{N}{\text{df}(w)}$$

where N is the number of documents and $\text{df}(w)$ the number containing w .

The second factor is the one worth noticing. It is a statement about the *collection*, not the document – the same weighting idea that PMI applies to word-context pairs. A term earns weight by being unevenly distributed.

A Collection of Six Documents

-
- D1 brew coffee filter best morning coffee
 - D2 espresso machine review espresso coffee compared
 - D3 laptop battery life screen brightness review
 - D4 tablet screen battery comparison review
 - D5 perfect morning cup coffee brew guide
 - D6 coffee coffee coffee coffee coffee coffee brew shop menu prices
hours location parking directions staff seating wifi music
decor opening closing weekend
-

Query: **coffee brew**

$df(\text{coffee}) = 4$, so $idf = \log_{10}(6/4) = 0.176$. $df(\text{brew}) = 3$, so $idf = \log_{10}(6/3) = 0.301$.

D6 is the interesting one. It is a real kind of document: a café page that mentions coffee constantly and answers no question at all.

Three Scorers, Three Answers

	len	tf(coffee)	tf(brew)	raw tf	tf-idf cos	BM25
D1	6	2	1	3	0.340	1.450
D2	6	1	0	1	0.046	0.502
D3	6	0	0	0	0.000	0.000
D4	5	0	0	0	0.000	0.000
D5	6	1	1	2	0.237	1.290
D6	22	6	1	7	0.247	1.096

raw tf	D6 > D1 > D5 > D2
tf-idf cosine	D1 > D6 > D5 > D2
BM25	D1 > D5 > D6 > D2

Raw counting hands the top slot to the café page. Every refinement after Luhn is, in one way or another, a defense against D6.

What the Cosine Fixes, and What It Misses

The cosine denominator normalizes by document length, which is why D6 falls to second. But the cosine still treats six occurrences as three times the evidence of two.

The intuition BM25 adds

The first occurrence of a term is strong evidence the document is about it. The sixth is nearly worthless. Relevance should **saturate**.

$$\text{BM25}(w, d) = \text{idf}(w) \cdot \frac{\text{tf}(w, d) (k_1 + 1)}{\text{tf}(w, d) + k_1 \left(1 - b + b \frac{|d|}{\text{avgdI}}\right)}$$

typically $k_1 \approx 1.2$ and $b = 0.75$

BM is *Best Matching*; the **25** is the twenty-fifth variant the group tried and means nothing else. From the Okapi system at City University London – Robertson, Spärck Jones and colleagues – established through the TREC evaluations of the 1990s. Unlike tf-idf it is derived from a probabilistic model of relevance rather than assembled from intuitions.

BM25: the Two Knobs

k_1 sets how fast relevance saturates.

tf	multiplier	gain
1	1.000	–
2	1.375	+0.375
3	1.571	+0.196
5	1.774	+0.203
10	1.964	+0.190
∞	2.200	

$k_1 = 0$ makes the term binary, present or absent. Large k_1 makes it nearly linear, like raw tf.

b sets the length penalty.

$ d /\text{avgdI}$	multiplier at tf = 2
0.5	1.600
1.0	1.375
2.0	1.073
4.0	0.746

$b = 0$ turns it off. $b = 1$ divides fully by relative length.

$b = 0.75$ concedes that some documents are long because they are thorough.

The whole per-term contribution is bounded by $\text{idf}(w) \cdot (k_1 + 1)$. The first occurrence earns a full unit, the second a third of that, and everything past ten is rounding error.

Saturation, in One Comparison

The contribution of coffee alone, under BM25:

	tf(coffee)	length	term score
D1	2	6	0.662
D6	6	22	0.676

Three times the term frequency buys an improvement of 0.014.

Saturation and length normalization together have almost exactly cancelled the repetition. That is the design working as intended: the function is asking *is this document about coffee*, and after two mentions it has already decided.

BM25 is thirty years old and remains a standard first-stage retriever, including in front of neural rankers. It is not a historical curiosity.

The Concordance: the Oldest Tool

Long before anything was computed, lexicographers read words in context by listing every occurrence with its neighbors. Keyword in context, KWIC.

<i>left context</i>	<i>node</i>	<i>right context</i>
a number to each individual, a	matrix	assigns a number to each ordered
indexed by one thing and a	matrix	by two, which is why a
a two-place relation becomes a	matrix	The pattern continues: three places would
the shortcut worth memorizing: multiplying a	matrix	by a one-hot vector selects
a one-hot row vector by a	matrix	selects a row. Nothing else happens
the entry is what the	matrix	assigns to that pair of

These are real lines, from a concordance of *matrix* over the text of this course's own slides – 159 occurrences in 41,860 tokens.

Read the node column downward and a pattern appears that no single sentence states: a *matrix assigns*, is *indexed by*, is what a relation *becomes*. Sinclair's argument for the COBUILD dictionary was that this is how word meaning should be described – from attested use, not from introspection.

Collocation: Firth's Question

"You shall know a word by the company it keeps." – Firth, 1957

Session 7 used that as a claim about *meaning*. Corpus linguistics uses it as a claim about *word combinations*: which pairs occur together more than chance allows?

The canonical pair:

strong tea fine

powerful tea **not English**, though the words mean nearly the same

No semantic theory predicts this. It is a fact about usage, recoverable only from counts, and it is exactly the kind of fact a learner of English needs and a dictionary should record.

Log-Likelihood, Defined

Build the 2×2 table for a bigram, over N bigram tokens:

	w_2	$\neg w_2$
w_1	$a = c(w_1, w_2)$	$b = c(w_1) - a$
$\neg w_1$	$c = c(w_2) - a$	$d = N - c(w_1) - c(w_2) + a$

Compare two hypotheses: that w_2 's probability does not depend on whether w_1 preceded it, against that it does. Twice the log of the likelihood ratio is

$$G^2 = 2 \sum_{ij} O_{ij} \ln \frac{O_{ij}}{E_{ij}}, \quad E_{ij} = \frac{(\text{row total})(\text{column total})}{N}$$

G^2 is asymptotically χ^2 with one degree of freedom, so it reads against a table: 3.84 at $p = .05$, 10.83 at $p = .001$. powerful tea scores 1.01 and fails; every other pair we will see passes.

Dunning, 1993. The point of the paper was that Pearson's χ^2 , which divides by E , becomes unreliable exactly where language data lives – cells with tiny expected counts. The likelihood ratio does not.

Two Statistics on the Same Counts

Take a corpus of 10^6 bigram tokens.

bigram	$c(w_1)$	$c(w_2)$	$c(w_1, w_2)$	PMI	LLR
of the	30000	60000	8000	2.15	13669.5
morning coffee	1200	900	55	5.67	330.3
strong tea	800	600	40	6.38	279.4
powerful tea	500	600	1	1.74	1.0
kumquat marmalade	3	5	3	17.61	75.6

by PMI	kumquat marmalade > strong tea > morning coffee > of the
by LLR	of the > morning coffee > strong tea > kumquat marmalade

powerful tea scores near zero on both, which is the result we wanted.

Why They Disagree

	PMI (Church & Hanks, 1990)	Log-likelihood (Dunning, 1993)
Asks	How surprising is this pairing, if the words were independent?	How much evidence is there that they are not independent?
Maximized by	Rare words that always co-occur	Frequent pairs, many observations
Fails on	Low counts: three occurrences of two rare words outscore any real collocation	Little, though it ranks of the first, rarely what a lexicographer wants

PMI = 17.61 for kumquat marmalade is computed from three observations. The statistic is not wrong; it answers a question about effect size with no regard for sample size.

Sparsity again – the same problem PPMI had with its zeros.

They Are One Quantity, Scaled

G^2 is not merely similar to mutual information. It *is* mutual information, times $2N$:

$$G^2 = 2 \sum_{ij} O_{ij} \ln \frac{O_{ij}}{E_{ij}} = 2N \sum_{ij} p_{ij} \ln \frac{p_{ij}}{p_{i\cdot} p_{\cdot j}} = 2N \cdot I(w_1; w_2)$$

	LLR	$2N \times I$	I (nats)
of the	13669.51	13669.51	0.006835
morning coffee	330.27	330.27	0.000165
strong tea	279.44	279.44	0.000140
kumquat marmalade	75.57	75.57	0.000038

So PMI and LLR are the *pointwise* and the *expected* mutual information of the same table, the second multiplied by the sample size. That is the entire difference between them.

Multiply every count and N by ten: kumquat marmalade goes from LLR 75.6 to 755.7 while its PMI stays at 17.61; strong tea goes from 279.4 to 2794.4 with PMI fixed at 6.38. PMI cannot see how many observations produced it; LLR is linear in them, which is also why LLR values do not compare across corpora of different sizes.

Word Sketches: Collocation by Relation

Raw collocation ignores structure. coffee near drink tells you less than knowing that coffee is the *object* of drink.

Kilgarriff's Word Sketches parse the corpus first, then count per grammatical relation:

object of	modifier	subject of
drink, brew, sip	black, strong, iced	spill, cool, taste

This is a word profile a lexicographer can write an entry from, and it is still counting – counting over typed relations instead of over a window.

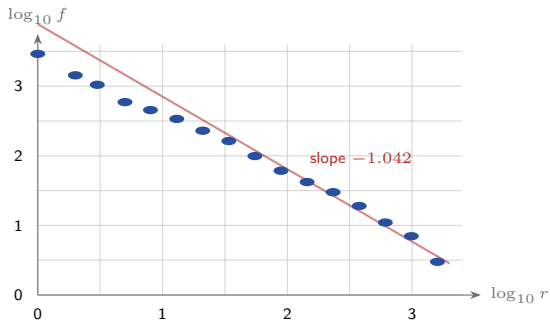
Note the shape of the move. Adding linguistic structure to the definition of “context” made the counts more informative. That is the same knob you turned in Session 7 when you chose a context window, and it is the knob prediction methods largely stopped turning.

Zipf, on This Course's Own Slides

r	word	f	$r \times f$
1	the	2913	2913
2	is	1431	2862
3	and	1046	3138
4	of	746	2984
5	it	590	2950
6	in	496	2976
7	to	457	3199
8	that	454	3632

41,860 tokens, 3,951 types. Fitted slope over ranks
1–2000: -1.042 .

The $r \times f$ column is nearly constant, which is Zipf's law stated as arithmetic rather than as a curve.



Why Sparsity Is Permanent

From the same corpus:

types occurring exactly once (<i>hapax legomena</i>)	1,389
as a share of the vocabulary	35.2%
as a share of the tokens	3.3%
share of tokens covered by the top 10 types	21.4%
by the top 100	48.0%
by the top 1000	84.9%

A third of the vocabulary was seen once. Heaps's law says this does not improve with scale: vocabulary grows as roughly T^β with β near 0.5, so a corpus ten times larger has a larger absolute hapax count, not a smaller one.

This is the structural reason the PPMI matrix was mostly zeros, and why no amount of additional data makes the zeros go away. **It is the problem every technique in this course is, in some sense, about.**

Question Answering by Counting

Who invented the telephone?

The hard approach parses the question, parses candidate documents, and reasons over the result. The AskMSR system (Brill, Dumais and Banko, 2002) did almost none of that.

The redundancy argument

On a large enough collection, the answer is stated somewhere in an easy form. Do not parse the hard sentence; find the easy one, and use counting to decide which candidate keeps appearing.

The procedure:

- 1 Rewrite the question into declarative patterns: “the telephone was invented by”
- 2 Send those as queries; collect the retrieved snippets
- 3 Harvest the n -grams adjacent to the matched pattern
- 4 Filter by expected answer type (here, a person name)
- 5 **Rank by frequency**

The Tally

candidate n -gram	count	
Alexander Graham Bell	14	returned as the answer
Antonio Meucci	3	
Elisha Gray	2	
Bell Labs	2	wrong type, filtered
the telephone	9	in the question, filtered

Illustrative counts over a set of retrieved snippets.

Nothing here understands the question. There is no parse, no inference, no representation of inventing or of telephones. There is a pattern, a filter, and a tally.

And on factoid questions it was competitive with systems that did all of those things. That result is genuinely uncomfortable, and it is worth sitting with rather than explaining away.

Where Redundancy Runs Out

Works when	the answer is a short string, stated often, in predictable phrasings
Fails when	the answer appears once, or must be assembled from several documents, or the question is not factoid
Fails badly when	the frequent string is not the true answer – popularity is not correctness

The Antonio Meucci row is the honest case. He has a real claim to the invention, and counting will never surface it, because counting measures how often something is said.

Note that this failure is not fixed by predicting instead of counting. A model trained on the same corpus inherits the same distribution of assertions. It is a property of the evidence, not of the method.

Language Models Were Counting Too

An n -gram language model estimates the next word from counts and nothing else:

$$P(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

Take a three-sentence corpus:

the cat sat on the mat | the dog sat on the rug | the cat ate the fish

bigram	$c(w_1, w_2)$	$c(w_1)$	MLE	add-one
the cat	2	6	0.333	0.176
cat sat	1	2	0.500	0.154
dog sat	1	1	1.000	0.167
dog ate	0	1	0.000	0.083

Vocabulary size 11, including sentence boundaries.

The Zero Problem, Again

the dog ate the fish is grammatical, and every word in it was observed. Its probability under the model is exactly zero, and its perplexity is infinite.

This is the espresso and cappuccino problem in a different costume. A count of zero means “not observed”, and the model reads it as “impossible”.

Add-one	Fixes the zero and wrecks the estimates: $P(\text{sat} \mid \text{dog})$ falls from 1.000 to 0.167
Good–Turing	Reallocate mass using the count of things seen exactly once – the hapax count again
Kneser–Ney	Back off to how many <i>distinct</i> contexts a word appears in, not how often

Two decades of smoothing research is one long attempt at a single question: what should a model believe about something it has never seen? The prediction methods answer it differently; they do not escape it.

The Lesson That Should Have Been a Warning

Brants and colleagues, at Google in 2007, trained n -gram models on two trillion tokens for machine translation.

Their smoothing method, *stupid backoff*, is close to the crudest thing that could work: if the n -gram is unseen, back off to the shorter one and multiply by a fixed constant. No normalization, not a probability distribution.

The finding

At sufficient scale, stupid backoff matched carefully engineered Kneser–Ney smoothing. The data mattered more than the estimator.

Read one way, this is a vindication of counting: just count more things. Read another way, it is the first clear signal that the returns were coming from scale rather than from linguistic insight.

Both readings turned out to be right, which is the uncomfortable part.

What Counting Does Well

Transparent	Every score decomposes into observations you can point at. When BM25 ranks a document first, you can say which terms did it
Cheap	One pass over the corpus, no training loop, no hyperparameter search worth the name
Incremental	Add a document, update the counts. No retraining
Well understood	Significance tests, confidence intervals, and known failure modes
Strong	BM25 is still a first-stage retriever; concordances still underlie dictionaries

Nothing in this list is true of a learned embedding, and some of it we will miss.

Where It Runs Out

Every method today shares one limitation, and it is the one Session 8 already showed you.

The limitation

A count-based method can only relate two things that were observed in the same place. If espresso and cappuccino never share a context, no counting statistic over those counts will relate them – because there is nothing there to count.

tf-idf, BM25	a query term missing from a document contributes nothing
PMI, LLR	a pair never observed has no statistic
n -gram models	an unseen bigram has probability zero

The SVD was the first serious answer: use the structure of the whole matrix to fill in what was never observed. It worked, and Session 8 showed exactly how fragile the result was.

The Pivot

The next move is to stop asking *how often did these co-occur* and start asking *how well can I predict one from the other*.

	Counting	Predicting
Object	a matrix of observations	parameters of a classifier
Zeros	absence of evidence	training signal, by sampling
Generalization	bolted on afterward, via the SVD	built into the objective
Dimensions	chosen by truncation	chosen in advance

Keep one question open as we go: word2vec looks like a different kind of object entirely. We will see later that, under a specific analysis, it is implicitly factoring a matrix of *shifted PMI values* – a matrix built from exactly the counts we have been discussing all along.

Levy and Goldberg, 2014. We will come back to it.

Exercises

- 1 D6 mentions coffee six times and still loses to D5, which mentions it once. Using the BM25 formula, identify which of the two mechanisms – saturation or length normalization – contributes more to that outcome.
- 2 kumquat marmalade has $PMI = 17.61$ from three observations. Compute what its PMI would be if all three counts were multiplied by 1000 and the corpus size were also multiplied by 1000. What does the answer tell you about what PMI measures?
- 3 A corpus has 35% hapax legomena. You double its size. Argue from Heaps's law whether the *proportion* of hapax types should rise, fall, or stay roughly constant.
- 4 The redundancy argument says the answer appears somewhere in an easy form. Name a question type for which this is false by construction, and say what a system would need instead.