Language Modeling and the Noisy Channel

Jan Hajič, Pavel Pecina, Jindřich Libovický

March 4, 2026

Charles University
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics

Outline

The Noisy Channel

Language Models

LM Smoothing and the EM Algorithm

The Noisy Channel

Outline

The Noisy Channel

Language Models

LM Smoothing and the EM Algorithm

Prototypical case:

	Input	**Channel**	**Output (noisy)**
	0,1,1,1,0,1,0,1,...	$\xrightarrow{\text{adds noise}}$	0,1,1,0,0,1,1,0,...

Model: probability of error (noise):

$$p(0|1) = 0.3 \quad p(1|1) = 0.7 \quad p(1|0) = 0.4 \quad p(0|0) = 0.6$$

The Task:

- *known:* the noisy output
- *want to know:* the input (decoding)

Noisy Channel Applications

- **OCR** — straightforward: text $\rightarrow$ print (adds noise), scan image $\rightarrow$ image
- **Handwriting recognition** — text $\rightarrow$ neurons, muscles ("noise"), scan/digitize $\rightarrow$ image
- **Speech recognition** (dictation, commands, ...) — text $\rightarrow$ conversion to acoustic signal ("noise") $\rightarrow$ acoustic waves
- **Machine Translation** — text in target language $\rightarrow$ translation ("noise") $\rightarrow$ source language

Nowadays, SoTA systems are end-to-end neural/LLM-based.

Bayes Rule for Noisy Channel

Recall Bayes Rule:

$$p(A|B) = \frac{p(B|A)\,p(A)}{p(B)}$$

The Golden Rule:

$$A_{\text{best}} = \arg\max_{A}\ p(B|A)\,p(A)$$

- $p(B|A)$: **acoustic/image/translation** application-specific name
- $p(A)$: the **language model**

Language Models

Outline

Probability of a Sentence

- Sequence of word forms: $W = (w_1, w_2, w_3, \ldots, w_d)$
- **The big modeling question:** $p(W) =?$
- By the chain rule (Bayes):

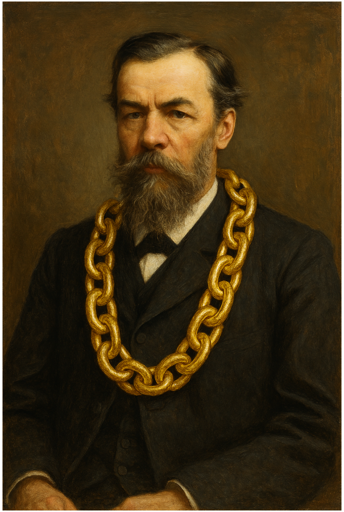
$$p(W) = p(w_1) \cdot p(w_2|w_1) \cdot p(w_3|w_1, w_2) \cdots p(w_d|w_1, \ldots, w_{d-1})$$

- Direct estimation is **not practical:** even short W requires too many parameters

Markov Chain

- **Unlimited memory** (cf. previous slide):
  - for w_i, we know all predecessors $w_1, w_2, \ldots, w_{i-1}$
- **Limited memory** (kth order Markov approximation):
  - disregard "too old" predecessors
  - remember only k previous words: $w_{i-k}, \ldots, w_{i-1}$
- **Stationary character** (no change over time):

$$p(W) \approx \prod_{i=1}^{d} p(w_i \mid w_{i-k}, w_{i-k+1}, \ldots, w_{i-1})$$



n-gram Language Models

$(n-1)$th order Markov approximation $\Rightarrow$ n-gram LM:

$$p(W) \stackrel{\text{def}}{=} \prod_{i=1}^{d} p(w_i \mid w_{i-n+1}, \ldots, w_{i-1})$$

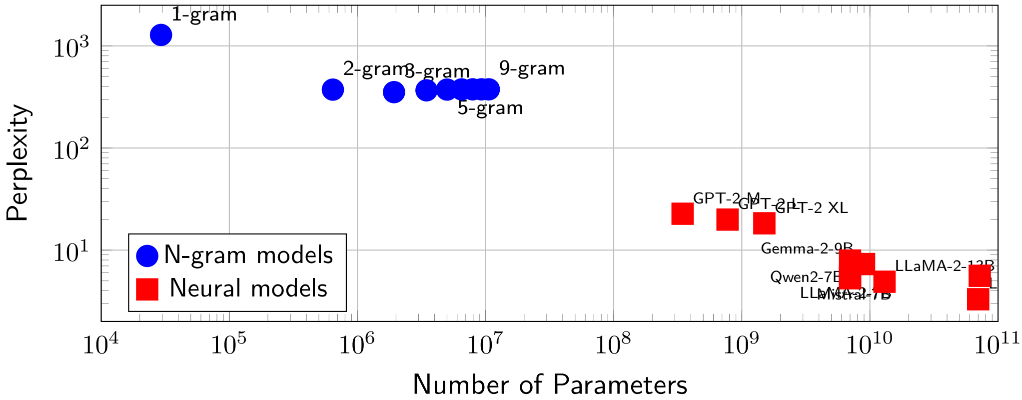
With vocabulary $|V| = 60\,000$:

Model	**Name**	**Parameters**
0-gram	uniform	1
1-gram	unigram	6×10^4
2-gram	bigram	3.6×10^9
3-gram	trigram	2.16×10^{14}

Will it be that many parameters in practice?

No, the count tables will be very sparse.

Number of parameters vs. perplexity on the Wiki



Not a fair comparison: with more data, the n-gram models would also slightly improve.

Maximum Likelihood Estimate

MLE = Relative Frequency — best predicts the training data T

For trigrams from training data T:

- Count $c_3(w_{i-2}, w_{i-1}, w_i)$ — sequences of three words in T
- Count $c_2(w_{i-1}, w_i)$ — sequences of two words in T
  - either $c_2(y, z) = \sum_w c_3(y, z, w)$
  - or count separately at beginning/end of data

MLE for trigrams:

$$\hat{p}(w_i \mid w_{i-2}, w_{i-1}) = \frac{c_3(w_{i-2}, w_{i-1}, w_i)}{c_2(w_{i-2}, w_{i-1})}$$

Training data:

`<s> <s> He can buy the can of soda.`

- **Unigram:** $p_1(\text{He}) = p_1(\text{buy}) = \cdots = 0.125; \quad p_1(\text{can}) = 0.25$
- **Bigram:** $p_2(\text{He}|\texttt{<s>}) = 1,\ p_2(\text{can}|\text{He}) = 1,\ p_2(\text{buy}|\text{can}) = 0.5, \ldots$
- **Trigram:** $p_3(\text{He}|\texttt{<s>},\texttt{<s>}) = 1,\ p_3(\text{can}|\texttt{<s>}, \text{He}) = 1, \ldots$
- **Entropy:** $H(p_1) = 2.75, \quad H(p_2) = 0.25, \quad H(p_3) = 0 \quad \rightarrow$ Great?!

No, the model is **overfitted** to the training data.

LM: An Example — The Problem

Cross-entropy on:

$S =$ `<s> <s> It was the greatest buy of all.`

Even $H_S(p_1) = \infty$ (and $= H_S(p_2) = H_S(p_3) = \infty$), because:

- all unigrams except $p_1(\text{the}), p_1(\text{buy}), p_1(\text{of}), p_1(.)$ are 0
- all bigram probabilities are 0
- all trigram probabilities are 0

Goal: make all (theoretically possible) probabilities non-zero.

Try our character-level LM as autocomplete!

https://ufallab.ms.mff.cuni.cz/~libovicky/ngram_predictor.html

LM Smoothing and the EM Algorithm

Outline

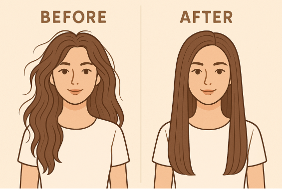
The Zero Problem

- "Raw" n-gram LM has necessarily some zeros
  - trigram model: $\sim 2.16 \times 10^{14}$ parameters; data $\sim 10^{9}$ words
- **Which are true zeros?**
  - Optimal: every trigram seen several times — impossible in practice
  - We don't know; we *must* eliminate the zeros
- **Two kinds of zeros:**
  - $p(w|h) = 0$ (history seen, word not)
  - $p(h) = 0$ (history itself unseen)

Why Do We Need Non-Zero Probabilities?

- **Avoid infinite cross-entropy:**
  - occurs when test data contains an event not seen in training
  - $H(p) = \infty$ prevents any meaningful comparison
- **Make the system more robust:**
  - Low-count estimates: detailed but less reliable
  - High-count estimates: reliable but less detailed

Smoothing — General Idea



Obtain new $p'(w)$ (same support): almost $p(w)$ but no zeros.

1. **Discount** some $p(w) > 0$: new $p'(w) < p(w)$

$$\sum_{w \text{ discounted}} \big(p(w) - p'(w)\big) = D$$

2. **Redistribute** D to all w with $p(w) = 0$ (possibly also to low-probability words)
3. Ensure $\sum_w p'(w) = 1$

Smoothing by Adding 1 (Laplace)

$$p'(w|h) = \frac{c(h,w) + 1}{c(h) + |V|}$$

Problem: if $|V| \gg c(h)$, too much probability mass is redistributed.

Example: Training: `<s> what is it what is small ?` $(|T| = 8, |V| = 12)$

- $p(\text{what}) = 0.25,\ p(\text{it}) = 0.125,\ p(.) = 0$
- $p'(\text{what}) \approx 0.15,\ p'(\text{it}) \approx 0.10,$
  $p'(.) \approx 0.05$

Adding Less Than 1 (δ-smoothing)

$$p'(w|h) = \frac{c(h,w) + \delta}{c(h) + \delta|V|}, \quad \delta \in (0,1)$$

Example with $\delta = 0.1$:

- p'(what) ≈ 0.23, p'(it) ≈ 0.12, p'(.) ≈ 0.01
- p'(it is flying.) ≈ 0.000003 (was 0 with raw model)

Linear Interpolation Smoothing

Combine distributions of various detail vs. reliability:

$$p'(w_i \mid w_{i-2}, w_{i-1}) = \lambda_3\, p_3 + \lambda_2\, p_2 + \lambda_1\, p_1 + \lambda_0\, \frac{1}{|V|}$$

with $\lambda_j > 0$ and $\sum_j \lambda_j = 1$.

- High λ_3: detailed but less reliable (sparse)
- High λ_1: reliable but less detailed

Estimating the λ Weights

If we use the training data, the best thing to do is to set $\lambda_3 = 1$ (MLE).
What shall we do?

- **Minimize cross-entropy on held-out data** H (not training data!):

$$-\frac{1}{|H|}\sum_{i=1}^{|H|}\log_2 p'(w_i|h_i)$$

- Why not training data? Using T always gives $\lambda_3 = 1$ (trivially minimized by MLE).
- Split data into: **training** T, **held-out** H, **test** S

The EM Algorithm for Smoothing

E-step — Expected counts:

$$c(\lambda_j) = \sum_{i=1}^{|H|} \frac{\lambda_j \, p_j(w_i|h_i)}{p'(w_i|h_i)}$$

M-step — Update:

$$\lambda_j^{\text{new}} = \frac{c(\lambda_j)}{\sum_k c(\lambda_k)}$$

Algorithm:

1. Start with all $\lambda_j > 0$
2. Compute expected counts (E-step)
3. Update λs (M-step)
4. Repeat until convergence: $|\lambda_j - \lambda_j^{\text{new}}| < \varepsilon$

Convergence guaranteed by Jensen's inequality.
$\log(\mathcal{E}[X]) \leq \mathcal{E}[\log(X)]$
$\rightarrow$ EM never decreases likelihood.

Simple EM Example

Unigram: $p(a) = 0.25,\ \ p(b) = 0.5,\ \ p(c \ldots r) = 1/64;$ **Heldout:** `baby`

Start: $\lambda_1 = 0.5$

$$p'(b) = 0.5 \times 0.5 + 0.5/26 \approx 0.27$$

Expected counts:

$$c(\lambda_1) \approx 2.72, \quad c(\lambda_0) \approx 1.28$$

Update: $\lambda_1^{\text{new}} \approx 0.68,\ \ \lambda_0^{\text{new}} \approx 0.32$

Repeat until $|\lambda_j - \lambda_j^{\text{new}}| < 0.01.$

Summary

- The **noisy channel model** underlies OCR, ASR, MT, and POS tagging; it requires a language model $p(A)$.
- n-gram LMs approximate $p(W)$ via a Markov assumption; estimated by MLE (relative frequency).
- Raw n-gram models suffer from **zero probabilities**; smoothing is essential.
- **Linear interpolation** with back-off to lower-order models is a principled solution.
- λ weights are estimated on **held-out data** using the **EM algorithm**.

https://ufal.mff.cuni.cz/courses/npfl124