Predicting the Next Word

From Shannon to ChatGPT

Test Compilation - Chapter 2

Contents

Chapter 2

N-gram Language Models

In this chapter, we advance next-word prediction by:

- Introducing the Markov assumption to make prediction tractable
- Estimating probabilities directly from word counts in corpora
- Developing smoothing techniques to handle unseen word sequences
- Understanding the fundamental trade-offs between context and data sparsity

2.1 N-grams and Next-Word Prediction

The fundamental challenge of next-word prediction lies in estimating conditional probabilities of the form $P(w_t \mid w_1, w_2, \ldots, w_{t-1})$, where the conditioning context can grow arbitrarily long as we proceed through a text. As we saw in Chapter **??**, naive estimation of these probabilities faces an exponential explosion: a vocabulary of just 50,000 words and contexts of length 10 would require storing $50,000^{10}$ parameters, a number exceeding the atoms in the observable universe. N-gram models provide the first practical solution to this challenge by making a bold simplifying assumption: the probability of the next word depends only on the preceding $n-1$ words, not the entire history. This **Markov assumption** transforms an intractable estimation problem into one that can be solved by counting occurrences in a training corpus. Despite their simplicity, n-gram models dominated natural language processing for over three decades, powering speech recognition systems, machine translation engines, spelling correctors, and countless other applications. Understanding n-gram models is essential not only for historical perspective but because they illuminate fundamental concepts—sparsity, smoothing, and the bias-variance trade-off—that remain central to modern neural language models.

The term "n-gram" refers to a contiguous sequence of n words from a text. A unigram is a single word, a bigram is a pair of consecutive words, a trigram is a triple, and so forth. When we speak of an "n-gram model," we mean a language model that conditions next-word predictions on the preceding $n-1$ words, using the statistics of n-grams observed in training data. Figure 2.1 illustrates this as a Markov chain where states represent words and transition probabilities capture the likelihood of moving from one word to another. The Markov property states that the future is conditionally independent of the past given the present: knowing the current state provides all the information needed to predict the next state, making earlier history irrelevant. In the context of language modeling, this means that once we know the last $n-1$ words, learning about words even further back provides no additional predictive power—a strong assumption that is clearly false for natural language but nevertheless proves useful as a computational approximation. The chain structure also reveals that n-gram models define a generative process: we can generate text by starting from a special beginning-of-sentence token and repeatedly sampling from the conditional distribution until we reach an end-of-sentence token, with each sampled word depending only on its immediate predecessors.

The choice of n represents a fundamental trade-off in language modeling that we will encounter repeatedly

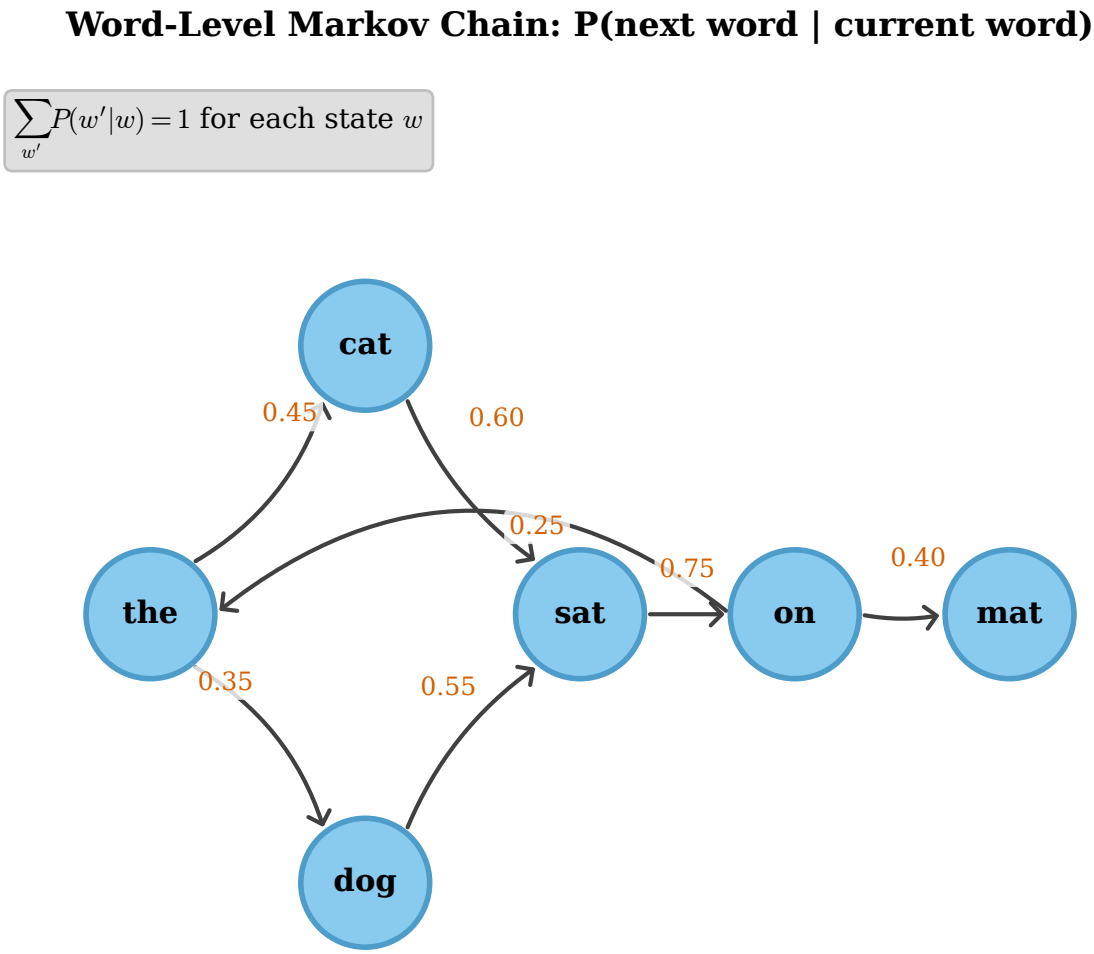

Figure 2.1: A word-level Markov chain illustrating n-gram dependencies. Each node represents a word state, and edges show transition probabilities between consecutive words. The probabilities on outgoing edges from each state sum to one, ensuring a valid probability distribution. This visualization captures the essence of bigram models where predictions depend only on the immediately preceding word.

throughout this book. Larger values of n capture more context, potentially improving prediction accuracy by considering longer-range dependencies in the text. A trigram model can distinguish between "New York" and "New Jersey" where a bigram model seeing only "New" cannot; a 5-gram model can capture idiomatic phrases like "at the end of the day" that shorter models would miss. However, larger n values come with severe costs: the number of possible n-grams grows exponentially with n, as there are $|\mathcal{V}|^n$ possible sequences of length n for a vocabulary $\mathcal{V}$. This exponential growth means that most n-grams will never appear in any finite training corpus, creating the **sparsity problem** that dominated n-gram research for decades. When we encounter a word sequence never seen in training, the raw count-based probability estimate is zero, which causes catastrophic failures when computing sequence probabilities since a single zero factor makes the entire product zero. The entire field of smoothing techniques arose to address this challenge, redistributing probability mass from seen events to unseen events in principled ways that we will explore in detail in Section 2.4.

Figure 2.2 visualizes how different n-gram orders define progressively larger context windows for next-

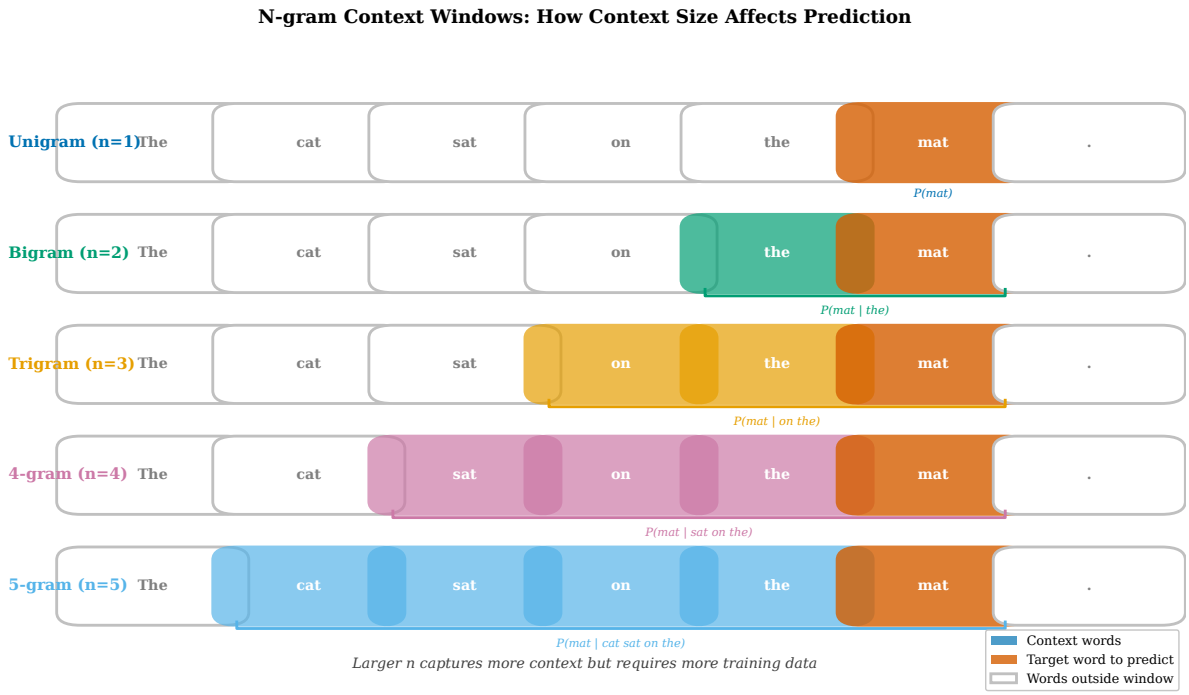

Figure 2.2: N-gram context windows from unigram through 5-gram. The target word to predict is highlighted in red, while context words are shown in the corresponding n-gram color. Larger n-gram orders capture more context but require exponentially more training data to estimate reliably. The probability notation shows how the conditioning set grows with n.

word prediction. The unigram model ignores context entirely, predicting words based solely on their overall frequency in the training corpus—"the" is predicted with high probability regardless of what preceded it, simply because "the" is the most common word in English. This context-free approach captures the basic distribution of words but misses all sequential structure. Bigram models condition on exactly one preceding word, capturing basic sequential dependencies like the tendency for articles to precede nouns, verbs to precede their objects, and prepositions to precede noun phrases. This single word of context provides substantial predictive power: knowing the previous word was "the" dramatically changes predictions compared to knowing it was "quickly." Trigram models extend this to two preceding words, capturing phrase-level patterns and disambiguating cases where single-word context is insufficient. Each additional word of context provides more information about the likely continuation but requires correspondingly more training data to estimate the resulting probabilities reliably. The growth in required data is not merely linear but combinatorial, as longer contexts create exponentially more possible patterns to estimate. In practice, n-gram models with $n > 5$ rarely improve performance because the contexts become so specific that they almost never repeat between training and test data, making the additional context effectively useless even though it theoretically contains valuable information about the intended continuation.

The historical dominance of n-gram models from the 1980s through the 2010s reflects their remarkable practical utility despite their theoretical limitations. At IBM, researchers developed language models that transformed speech recognition from a curiosity into a practical technology by combining acoustic models with n-gram language models that favored linguistically plausible word sequences [Jelinek, 1990]. When the acoustic signal was ambiguous between "recognize speech" and "wreck a nice beach," the language model strongly favored the former, dramatically improving transcription accuracy. Similar success followed in statistical machine translation, where target-language n-gram models guided the decoder toward fluent output even when the translation model produced awkward word-by-word correspondences. Spelling correction, text classification, authorship attribution, and information retrieval all benefited from n-gram language models during this golden age of statistical NLP. Figure 2.3 traces this history, showing how n-gram models evolved from theoret-

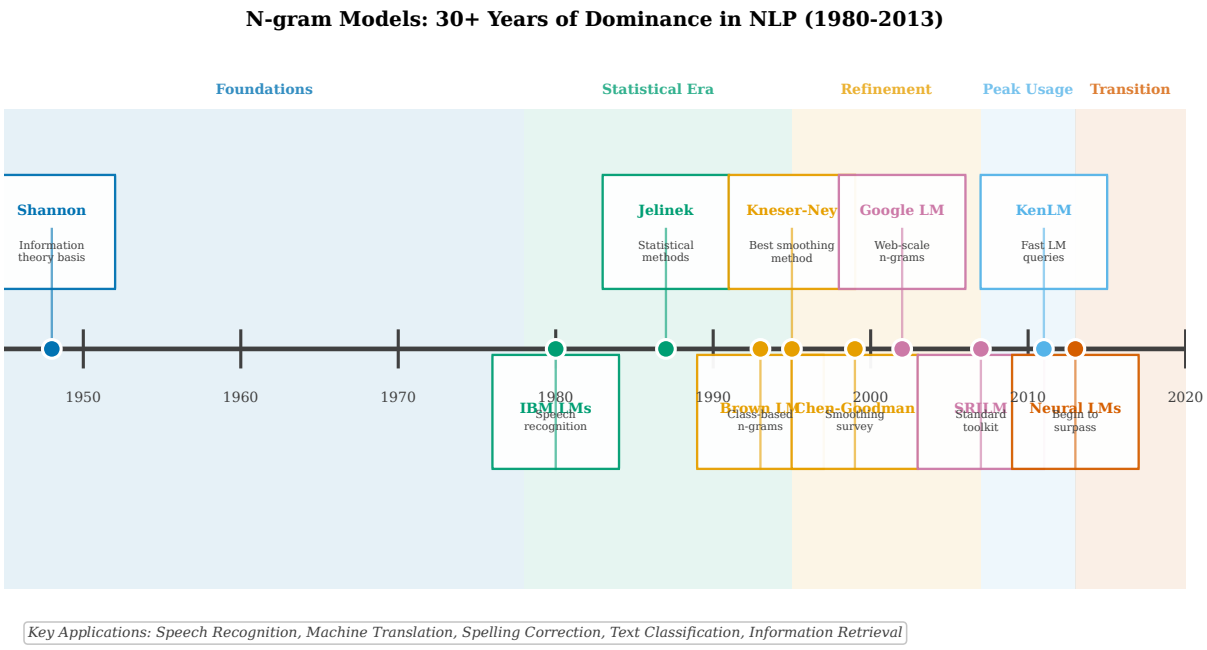

Figure 2.3: Historical impact of n-gram models in NLP from 1948 to 2013. Key milestones include Shannon's information-theoretic foundations, IBM's speech recognition systems, the development of sophisticated smoothing techniques, and the eventual transition to neural approaches. N-gram models dominated practical NLP for over three decades.

ical curiosity to practical necessity before eventually yielding to neural approaches that could overcome their fundamental limitations.

The transition from n-gram to neural language models beginning around 2013 did not render n-gram concepts obsolete; rather, it provided new tools to address the same fundamental challenges that have always confronted language modeling. The sparsity problem that motivated decades of n-gram smoothing research reappears in neural models as the challenge of generalizing from training data to unseen inputs—neural networks must also learn to handle novel word combinations not explicitly observed during training. The bias-variance trade-off between short contexts that estimate reliably but predict poorly versus long contexts that could predict well but estimate unreliably finds its neural analog in the trade-off between model capacity and overfitting: larger neural networks can capture more complex patterns but risk memorizing training data rather than learning generalizable rules. The principle that we should back off from sparse contexts to more reliable shorter contexts anticipates the hierarchical representations that deep neural networks learn, where earlier layers capture local patterns like character n-grams and morphology while later layers integrate global context spanning entire sentences or documents. Even the specific techniques matter: the continuation probability insight from Kneser-Ney smoothing—that word versatility matters more than raw frequency—foreshadows the distributional semantics that neural word embeddings capture. Understanding n-gram models thus provides not merely historical context but conceptual foundations that illuminate the design of modern architectures and the challenges they continue to face in predicting the next word.

2.2 The Markov Assumption and Maximum Likelihood Estimation

The mathematical foundation of n-gram language models rests on two pillars: the chain rule of probability, which decomposes joint probabilities into products of conditionals, and the Markov assumption, which truncates these conditionals to depend only on recent history. To compute the probability of a sentence $w_1, w_2, \ldots, w_T$, we apply the chain rule to factor the joint probability as $P(w_1, w_2, \ldots, w_T) = P(w_1) \cdot P(w_2|w_1) \cdot P(w_3|w_1, w_2) \cdots P(w_T|w_1, \ldots, w_{T-1})$. This decomposition is mathematically exact—it follows directly from the definition of conditional probability and holds for any probability distribution over sequences—but the condi-

tioning contexts grow without bound as we proceed through the sentence, making direct estimation from finite data impossible. With a vocabulary of 50,000 words, estimating $P(w_{100}|w_1,\ldots,w_{99})$ would require observing the same 99-word prefix multiple times, which essentially never happens. The k-th order Markov assumption resolves this intractability by asserting that $P(w_t \mid w_1,\ldots,w_{t-1}) \approx P(w_t \mid w_{t-k},\ldots,w_{t-1})$: the probability of the next word depends only on the preceding k words, regardless of what came before. For bigram models ($k = 1$), this means $P(w_t \mid w_1,\ldots,w_{t-1}) \approx P(w_t \mid w_{t-1})$, reducing the infinite-history problem to one requiring only pairwise word statistics that appear frequently enough to estimate reliably. This assumption is linguistically naive—long-range dependencies abound in natural language, from pronoun resolution to discourse coherence—but it transforms an intractable estimation problem into one that can be solved by the simple operation of counting co-occurrences in a training corpus.

Once we adopt the Markov assumption, the question becomes how to estimate the required conditional probabilities from a training corpus. The principle of **Maximum Likelihood Estimation** (MLE) provides the answer: choose parameter values that maximize the probability the model assigns to the observed training data. For n-gram models, this principle leads to a remarkably simple recipe: the MLE estimate of a conditional probability equals the ratio of counts. For bigrams, $\hat{P}_{\text{MLE}}(w_2|w_1) = \frac{\text{count}(w_1,w_2)}{\text{count}(w_1)}$, where $\text{count}(w_1,w_2)$ is the number of times the bigram (w_1,w_2) appears in training data and $\text{count}(w_1)$ is the number of times w_1 appears. This formula has intuitive appeal: if "the cat" appears 847 times and "the" appears 3,412 times, we estimate $P(\text{cat}|\text{the}) = 847/3412 \approx 0.248$. The MLE principle can be derived by writing the likelihood function as a product over all training bigrams, taking the logarithm to convert products to sums, and differentiating with respect to the probability parameters subject to the constraint that probabilities sum to one. The result is the count ratio formula, which emerges as the unique maximizer of the training data likelihood.

Figure 2.4 provides a three-dimensional visualization of bigram probabilities for a small vocabulary, offering geometric intuition for the statistical structure that n-gram models capture. The dramatic height variations across the surface reveal the non-uniform structure of natural language: certain word transitions are highly probable while most are vanishingly rare, creating a landscape of sharp peaks rising from a nearly flat plain. The tall peaks correspond to grammatically natural sequences—articles followed by nouns, nouns followed by verbs, verbs followed by prepositions, and conjunctions followed by pronouns—while the flat regions near zero represent word pairs that rarely or never co-occur, such as consecutive articles or prepositions followed by punctuation. This visualization makes concrete the linguistic regularities that n-gram models capture: despite treating words as atomic symbols with no understanding of meaning or grammar, the statistical patterns of co-occurrence encode substantial information about language structure. The model has no concept of "noun" or "verb" as grammatical categories, yet the probability surface implicitly reflects these categories through the patterns of which words follow which. The sparse nature of the surface, with most values near zero and occasional tall peaks, also foreshadows the sparsity problem that dominates n-gram research: as vocabulary size grows from tens to thousands to tens of thousands, the proportion of zero-probability cells increases rapidly toward 100%, and we must develop sophisticated techniques to assign non-zero probability to the unseen word pairs that inevitably appear in test data.

The count matrix underlying MLE estimation reveals the severity of the sparsity problem. Figure 2.5 displays both the raw bigram counts and a binary visualization of observed versus unobserved pairs. Even with a vocabulary of just ten words, many cells in the matrix contain zeros—word pairs that never appeared in training data. The logarithmic color scale in the left panel shows counts spanning several orders of magnitude, from single occurrences to thousands, while the right panel starkly divides the matrix into seen (non-zero) and unseen (zero) cells. The sparsity statistics are sobering: even this tiny vocabulary exhibits significant gaps in coverage. For realistic vocabularies of 50,000 or more words, the matrix contains $|\mathscr{V}|^2 = 2.5 \times 10^9$ cells, and even massive training corpora of billions of words cannot hope to observe all possible bigrams. The situation worsens dramatically for higher-order n-grams: trigram models have $|\mathscr{V}|^3$ possible contexts, 4-gram models have $|\mathscr{V}|^4$, and so on. This exponential growth in the parameter space relative to the polynomial growth of training data is the fundamental mathematical obstacle that n-gram models must overcome.

Figure 2.6 traces the complete MLE pipeline from corpus to probability estimates. The training sentences yield bigram counts through a simple enumeration: we slide a window of width two across each sentence, incrementing the count for each observed word pair. The count table aggregates these observations, showing how

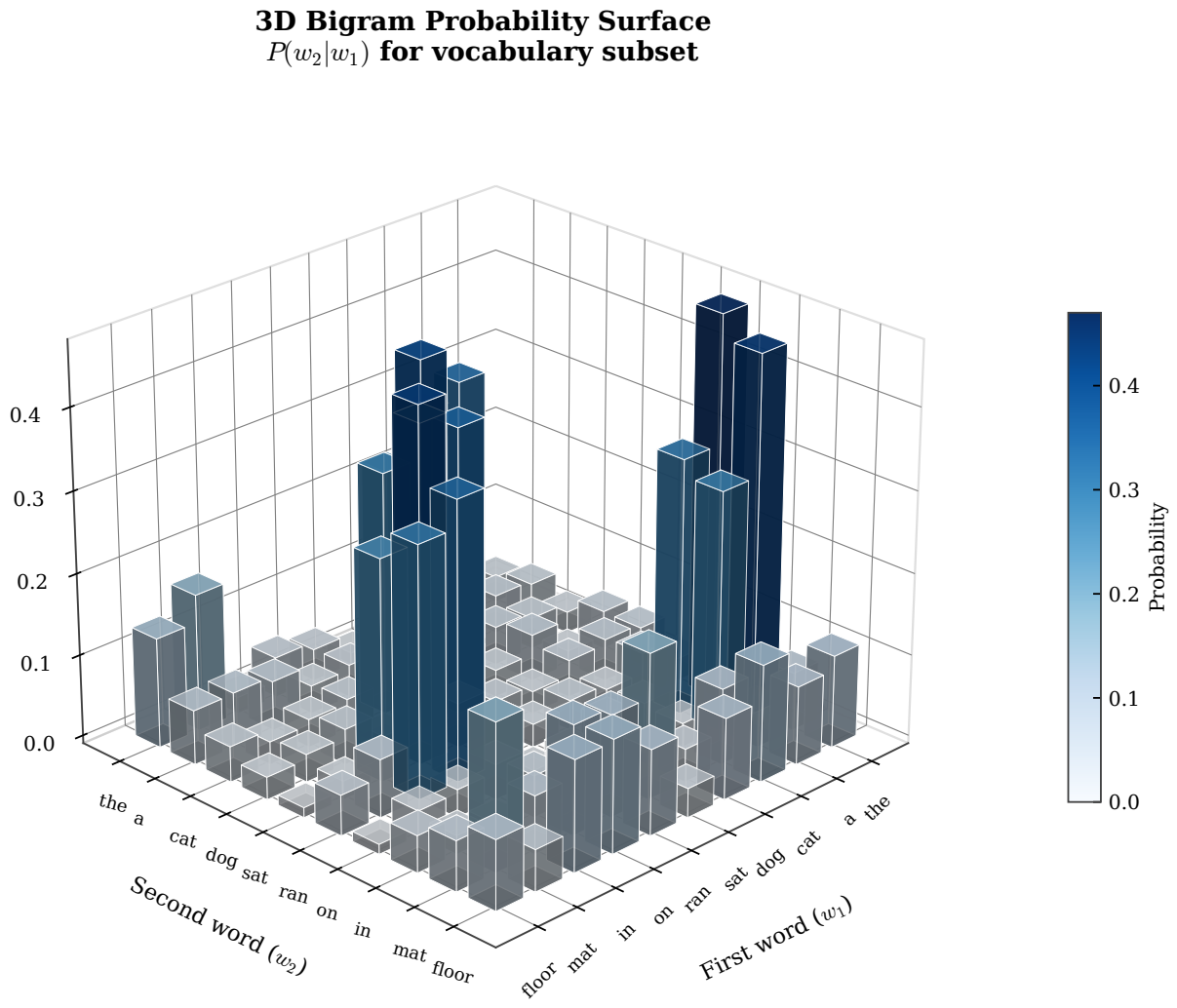

Figure 2.4: 3D visualization of bigram probabilities $P(w_2|w_1)$ for a vocabulary subset. Each bar represents the conditional probability of the second word given the first. The height variations reveal linguistic structure: articles precede nouns, nouns precede verbs, and verbs precede prepositions with high probability. Most word pairs have near-zero probability, illustrating the sparsity inherent in natural language.

frequently each bigram appears. The normalization step converts counts to probabilities by dividing each bigram count by the count of its first word, ensuring that the conditional distribution $P(\cdot|w_1)$ sums to one for each conditioning word w_1. The example calculations make the arithmetic concrete: $P(\text{sat}|\text{cat}) = 2/3 \approx 0.67$ means that two-thirds of the time we see "cat" in training, it is followed by "sat." The key insight box emphasizes MLE's Achilles' heel: if a bigram never appears in training, its count is zero, and dividing zero by any positive number yields zero. This means MLE assigns probability zero to any word sequence containing an unseen bigram, which is catastrophic when evaluating test data that inevitably contains novel word combinations.

The consequences of zero probabilities extend beyond individual bigrams to entire sentences. Consider computing the probability of a sentence using the chain rule: $P(w_1, \ldots, w_T) = \prod_{t=1}^{T} P(w_t|w_{t-1})$. If even a single factor in this product is zero, the entire sentence probability becomes zero, regardless of how well the model predicted the other words. This means that a single unseen bigram—perhaps a newly coined term, a rare proper name, or an unusual but grammatical construction—causes the model to assign zero probability to an entire sentence, text, or document. When used in applications like speech recognition or machine translation, this manifests as the model declaring certain outputs impossible rather than merely improbable. The problem worsens when we compute perplexity, the standard evaluation metric for language models, which involves

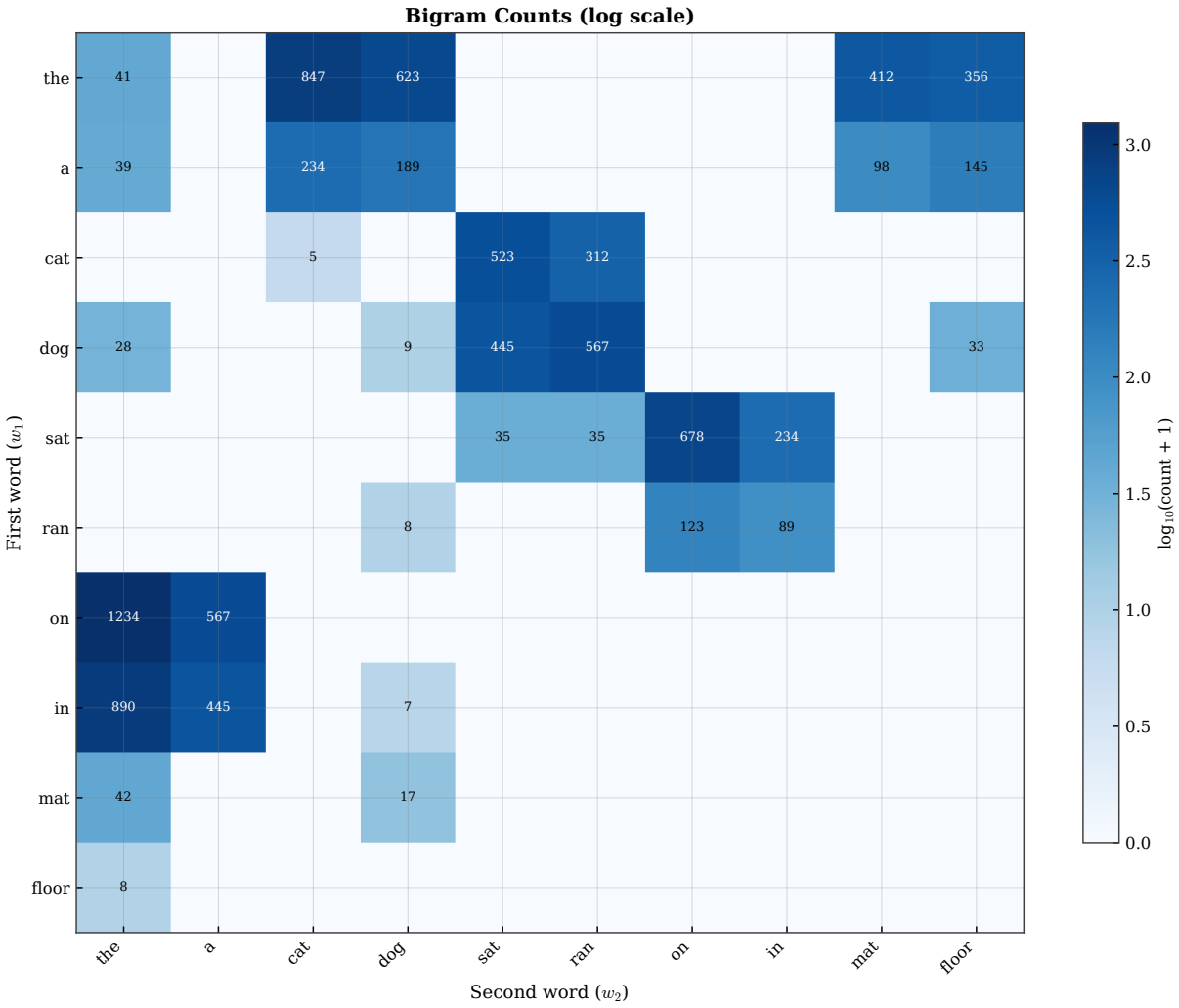

Figure 2.5: Bigram count matrix visualization. (a) Raw counts on a logarithmic scale, showing the wide dynamic range from rare to common bigrams. (b) Binary observed/unobserved pattern, revealing that most potential bigrams never appear in training data. The sparsity statistics quantify the severity of this problem even for small vocabularies.

taking the geometric mean of inverse probabilities: if any probability is zero, the perplexity becomes infinite, making model comparison meaningless. Smoothing techniques, which we explore in Section 2.4, address this problem by redistributing probability mass from observed events to unobserved events, ensuring that no n-gram receives exactly zero probability while still favoring patterns observed in training.

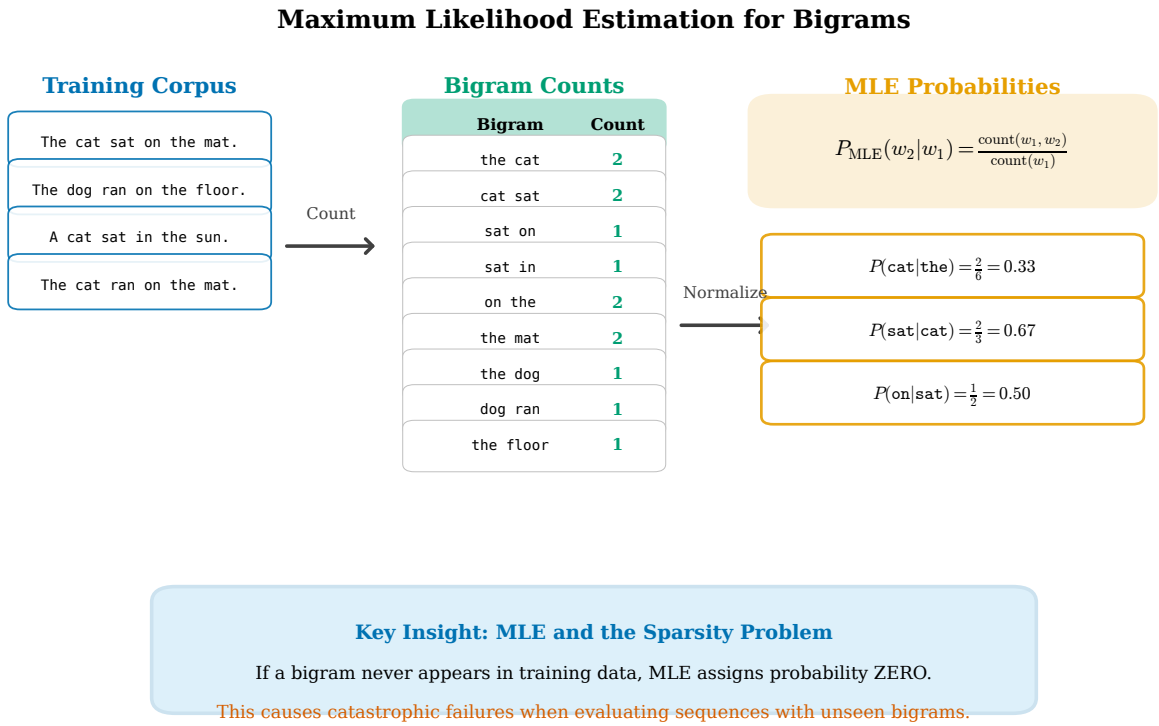

Figure 2.6: Maximum Likelihood Estimation process for bigram probabilities. The training corpus is processed to extract bigram counts, which are then normalized by unigram counts to produce probability estimates. Example calculations show how count ratios yield conditional probabilities. The key insight highlights MLE's critical weakness: unseen bigrams receive zero probability.

2.3 The Sparsity Problem

The fundamental obstacle facing n-gram language models is **sparsity**: the vast majority of possible word sequences never appear in any finite training corpus, yet many of these unseen sequences are perfectly valid and will inevitably occur in test data. This problem is not merely technical but mathematical: the number of possible n-grams grows exponentially with the vocabulary size and n-gram order, while the amount of training data we can collect grows at best polynomially. For a vocabulary of $|\mathcal{V}| = 50{,}000$ words, the number of possible bigrams is $|\mathcal{V}|^2 = 2.5$ billion, the number of possible trigrams is $|\mathcal{V}|^3 = 125$ trillion, and the number of possible 4-grams exceeds 6×10^{18}. Even the largest text corpora containing billions of words cannot hope to observe more than a tiny fraction of these possibilities. The consequence is stark: raw MLE estimates assign zero probability to almost all word sequences, making the model useless for evaluating new text that contains even a single unseen n-gram. Understanding the sparsity problem in depth is essential because the entire edifice of smoothing techniques arose specifically to address this challenge.

Figure 2.7 illustrates the catastrophic nature of zero probabilities with a concrete example. Consider evaluating the perfectly reasonable sentence "The cat sat on the velvet cushion" using a bigram model trained on some corpus. If the bigram "velvet cushion" never appeared in training—perhaps because the corpus lacked interior decorating content—the MLE estimate assigns $P(\text{cushion}|\text{velvet}) = 0$. When we compute the sentence probability using the chain rule, this zero propagates through the product: $P(\text{sentence}) = 0.15 \times 0.25 \times 0.40 \times 0.45 \times 0.12 \times 0.00 \times \ldots = 0$. The model declares the sentence impossible, assigning it the same probability as genuine gibberish like "The the sat velvet on cushion." This behavior is clearly wrong—the sentence is grammatical and meaningful—but it follows inevitably from MLE's reliance on observed counts. The problem worsens with higher-order n-grams: trigram models require seeing specific three-word sequences, 4-gram models require four-word sequences, and so on. The more context we include, the more specific the required

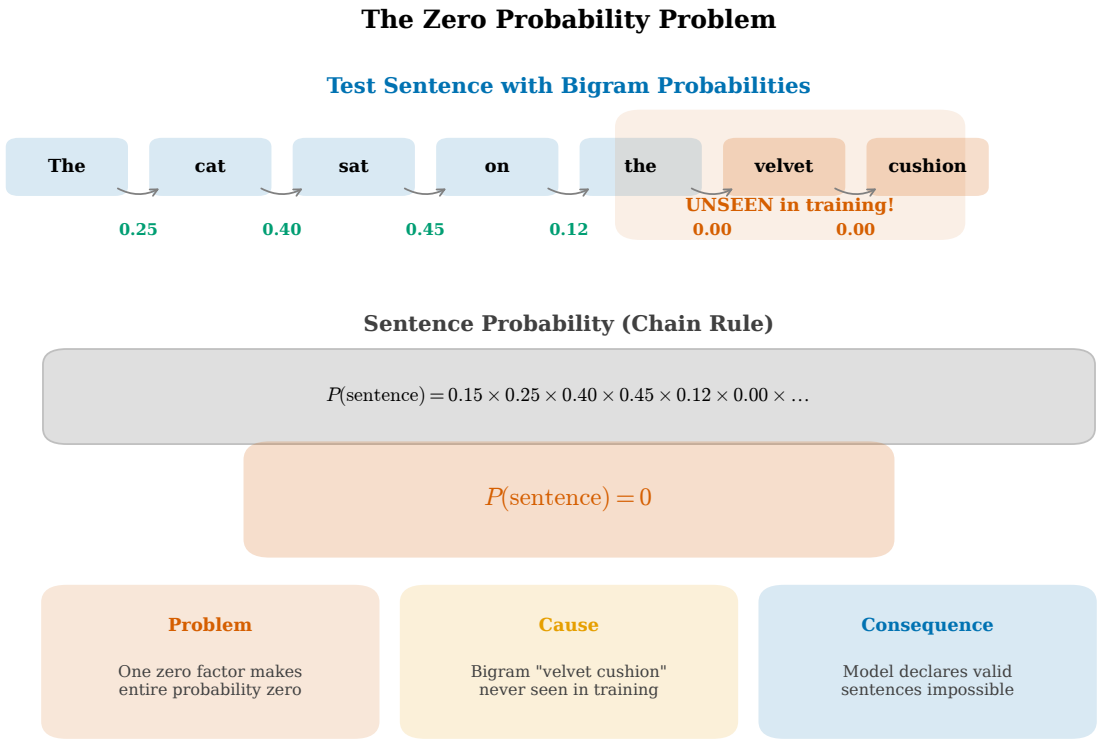

Figure 2.7: The zero probability problem in action. A single unseen bigram ("velvet cushion") causes the entire sentence probability to become zero when using MLE estimation. The chain rule multiplies all bigram probabilities, so one zero factor annihilates the entire product regardless of how well the model predicted other transitions.

observations become, and the less likely we are to have seen exactly what we need.

The mathematical structure of sparsity becomes clear when we examine how the parameter space grows relative to available data. Figure 2.8 quantifies this mismatch between model capacity and data availability. The left panel shows the exponential growth of possible n-grams: with a 50,000-word vocabulary, bigrams number in the billions, trigrams in the trillions, and 4-grams in the quintillions. These numbers dwarf even the largest available corpora. The right panel shows the consequence: as we add more training data, the fraction of possible n-grams we observe increases, but the growth is sublinear—doubling our corpus does not double our coverage. For trigrams and higher orders, even corpora of billions of words observe only a minuscule fraction of the possible parameter space. This creates a fundamental barrier: no amount of data collection can overcome the exponential growth in model parameters. The only solution is to share statistical strength across similar contexts, which is precisely what smoothing techniques accomplish.

The coverage curves in Figure 2.9 reveal the practical implications of sparsity. The left panel shows that adding more training data yields diminishing returns: the curve flattens as corpus size increases, indicating that new text increasingly contains n-grams we have already seen rather than novel ones. This behavior follows from Heaps' law, which states that vocabulary growth is sublinear in corpus size, and extends to n-grams of all orders. The right panel shows the frequency distribution of n-grams, revealing the long tail that characterizes natural language: a small number of common n-grams (the "head") account for most of the tokens in any corpus, while an enormous number of rare n-grams (the "long tail") appear only once or twice each. These singleton and doubleton n-grams, called hapax legomena and dis legomena respectively, pose a particular challenge: their low frequency means our count-based estimates are highly unreliable, yet they collectively account for a substantial portion of the vocabulary and cannot simply be ignored.

Figure 2.10 provides a three-dimensional perspective on how sparsity depends jointly on vocabulary size and n-gram order. The surface rises steeply from the lower-left corner (small vocabulary, bigrams) toward the

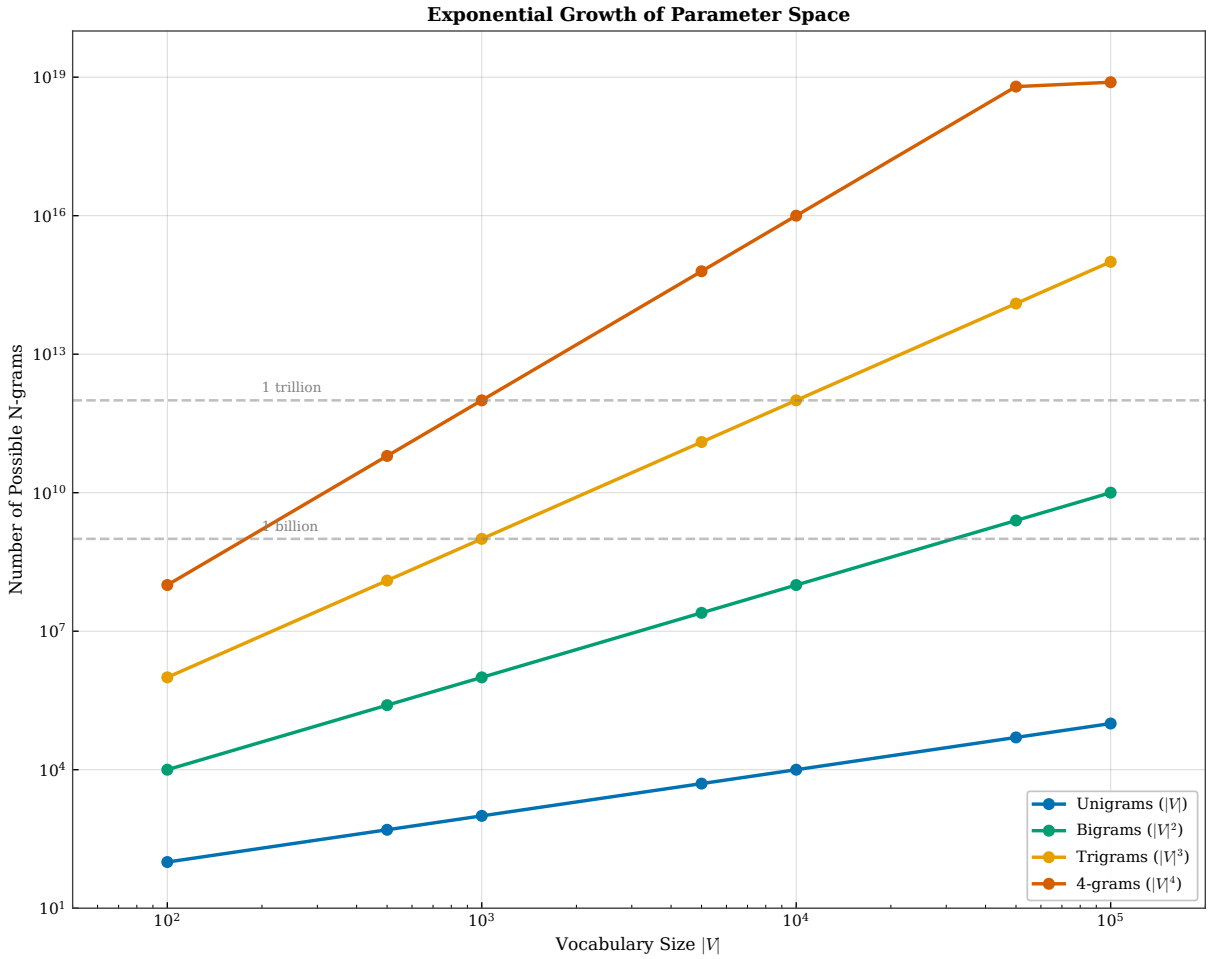

Figure 2.8: The curse of dimensionality in n-gram models. (a) The number of possible n-grams grows exponentially with vocabulary size and n-gram order, quickly exceeding trillions of possibilities. (b) Coverage of possible n-grams increases sublinearly with corpus size, meaning even billions of words observe only a small fraction of the parameter space.

upper-right (large vocabulary, higher-order n-grams), approaching 100% sparsity—meaning almost none of the possible n-grams have been observed. This visualization makes viscerally clear why practitioners rarely use n-grams beyond order 5: the parameter space becomes so vast relative to available data that the additional context provides no benefit. The surface also reveals an important interaction: increasing vocabulary size has a multiplicative effect with n-gram order, so the sparsity penalty for using a larger vocabulary is more severe for higher-order models. This creates pressure to limit vocabulary size through techniques like replacing rare words with a generic unknown token, trading model expressiveness for improved statistical reliability.

The statistical structure underlying sparsity is captured by Zipf's law, visualized in Figure 2.11. George Kingsley Zipf, a Harvard linguist, observed in the 1930s and 1940s that word frequencies in natural language follow a remarkably consistent power law: the frequency of a word is approximately inversely proportional to its rank when words are sorted by decreasing frequency, expressed mathematically as $f(r) \propto r^{-\alpha}$ where r is the rank and the exponent $\alpha \approx 1$ for most natural languages. The most common word in English ("the") appears thousands of times more frequently than words in the long tail of the distribution, dominating any corpus by sheer repetition while rare technical terms may appear only once. The left panel shows this relationship on a log-log scale, where a straight line indicates a power law relationship—the remarkable linearity spanning several orders of magnitude confirms that language statistics follow this pattern with surprising consistency across diverse corpora and genres. The right panel shows the cumulative distribution function: the top 100 most frequent words account for roughly 50% of all tokens in typical text, and a few thousand high-frequency

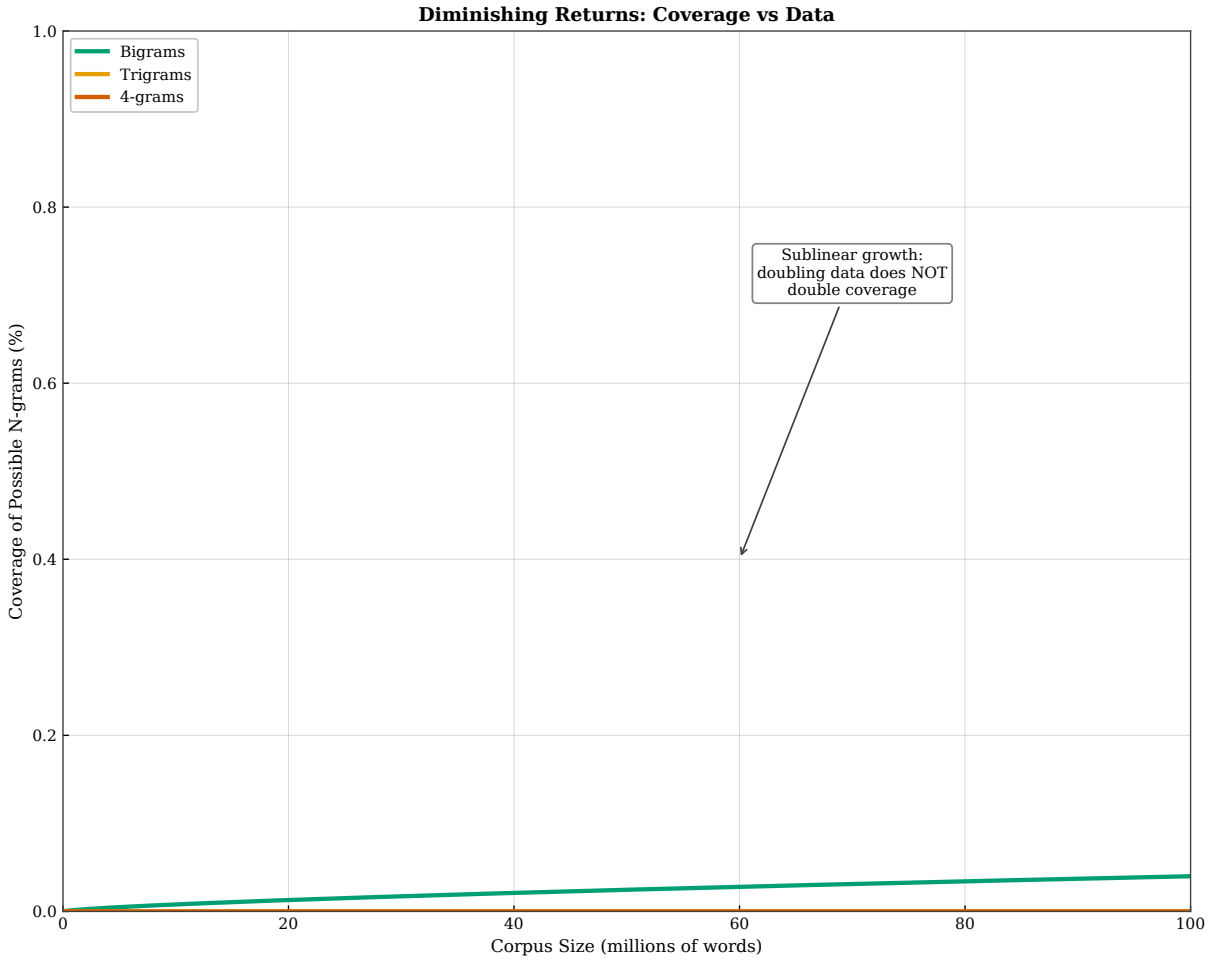

Figure 2.9: N-gram coverage patterns. (a) Coverage of possible n-grams grows sublinearly with corpus size, exhibiting severe diminishing returns. (b) The long-tail distribution of n-gram frequencies: a small number of n-grams account for most tokens, while the vast majority are rare, appearing only once or twice in even large corpora.

words cover 80-90% of the tokens we encounter in everyday reading. Yet that remaining 10-20% of tokens comprises tens of thousands of rare words, each essential for expressing specific technical concepts, proper names, and nuanced distinctions. This heavy-tailed distribution explains why n-gram sparsity is so mathematically severe: if individual words are rare, their combinations are exponentially rarer still, following the product of their individual probabilities. A bigram involving two rare words inherits the rarity of both components, and trigrams compound this effect cubically rather than linearly as each additional word multiplies the sparsity. The Zipfian distribution is not a peculiarity of English but appears across all documented human languages and even constructed languages like Esperanto, strongly suggesting it emerges from fundamental properties of efficient communication systems that balance speaker effort (favoring reuse of common words) against listener comprehension (requiring precision through specialized vocabulary for specific meanings).

The sparsity problem motivates the entire field of smoothing, which we develop in the following section as the central technical contribution of n-gram language modeling research. The key insight is that while we cannot observe all possible n-grams in any finite corpus, we can use what we have observed to make educated guesses about what we have not observed. If we have seen “the blue car” and “the red car” but never “the green car,” we might reasonably infer that “the green car” is plausible by analogy—colors are interchangeable in this construction, suggesting that any color could follow “the” before “car.” Smoothing techniques formalize this intuition in mathematical terms, redistributing probability mass from observed events to similar unobserved events according to various principles. The simplest approaches add small pseudo-

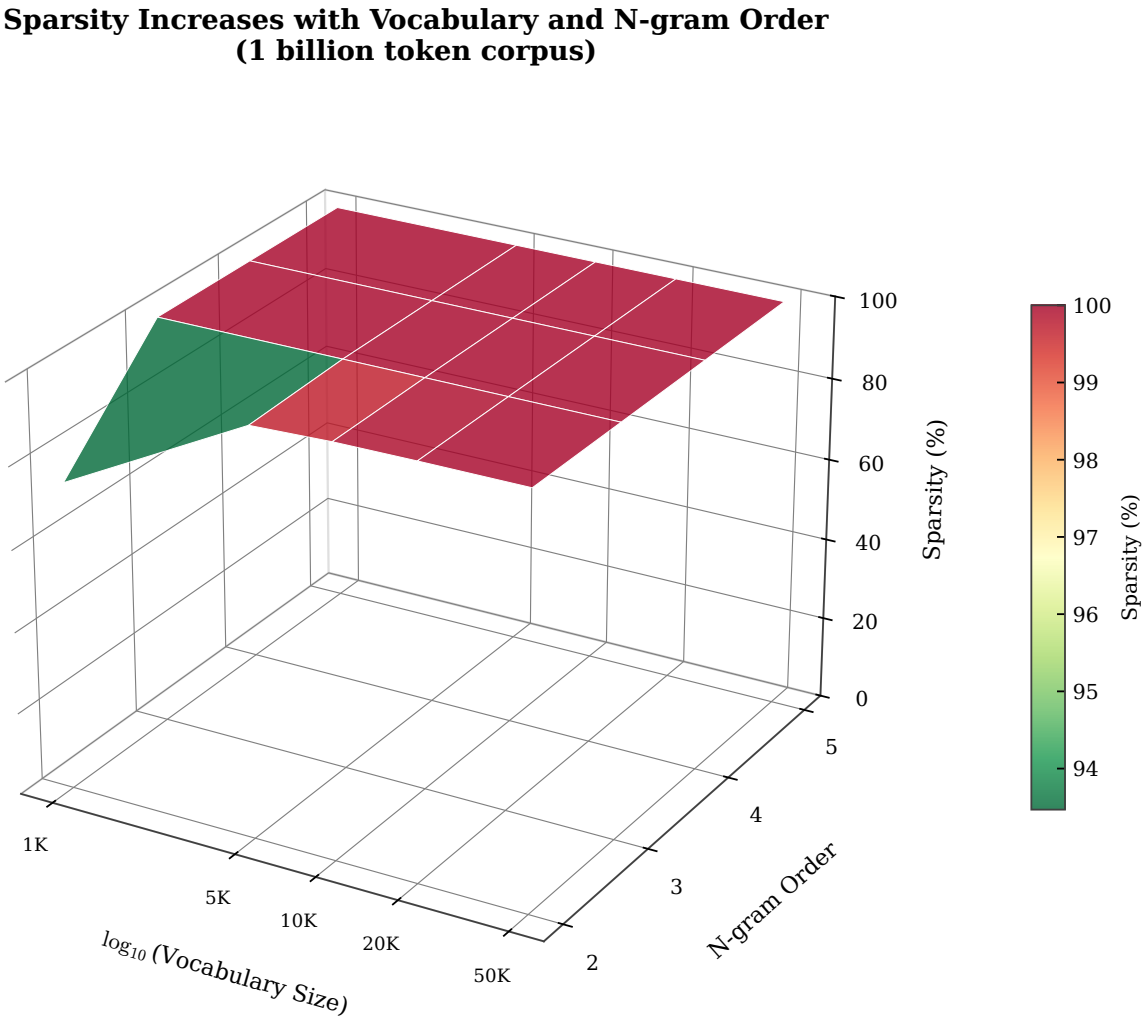

Figure 2.10: 3D visualization of sparsity as a function of vocabulary size and n-gram order. The surface rises steeply toward 100% sparsity as either dimension increases. Even with a billion-token corpus, higher-order n-grams with realistic vocabularies exhibit near-total sparsity, with almost all possible sequences unobserved.

counts to all n-grams regardless of whether they were observed (Laplace smoothing), treating all unseen events as equally likely. More sophisticated methods like Good-Turing estimation use the statistics of rare events to estimate how much mass should go to unseen events, while Kneser-Ney smoothing incorporates information about word versatility to distribute mass more intelligently. The common thread across all these methods is accepting that zero counts do not mean zero probability—they simply mean we lack direct evidence and must rely on indirect statistical inference to estimate what we would likely see if we had more data.

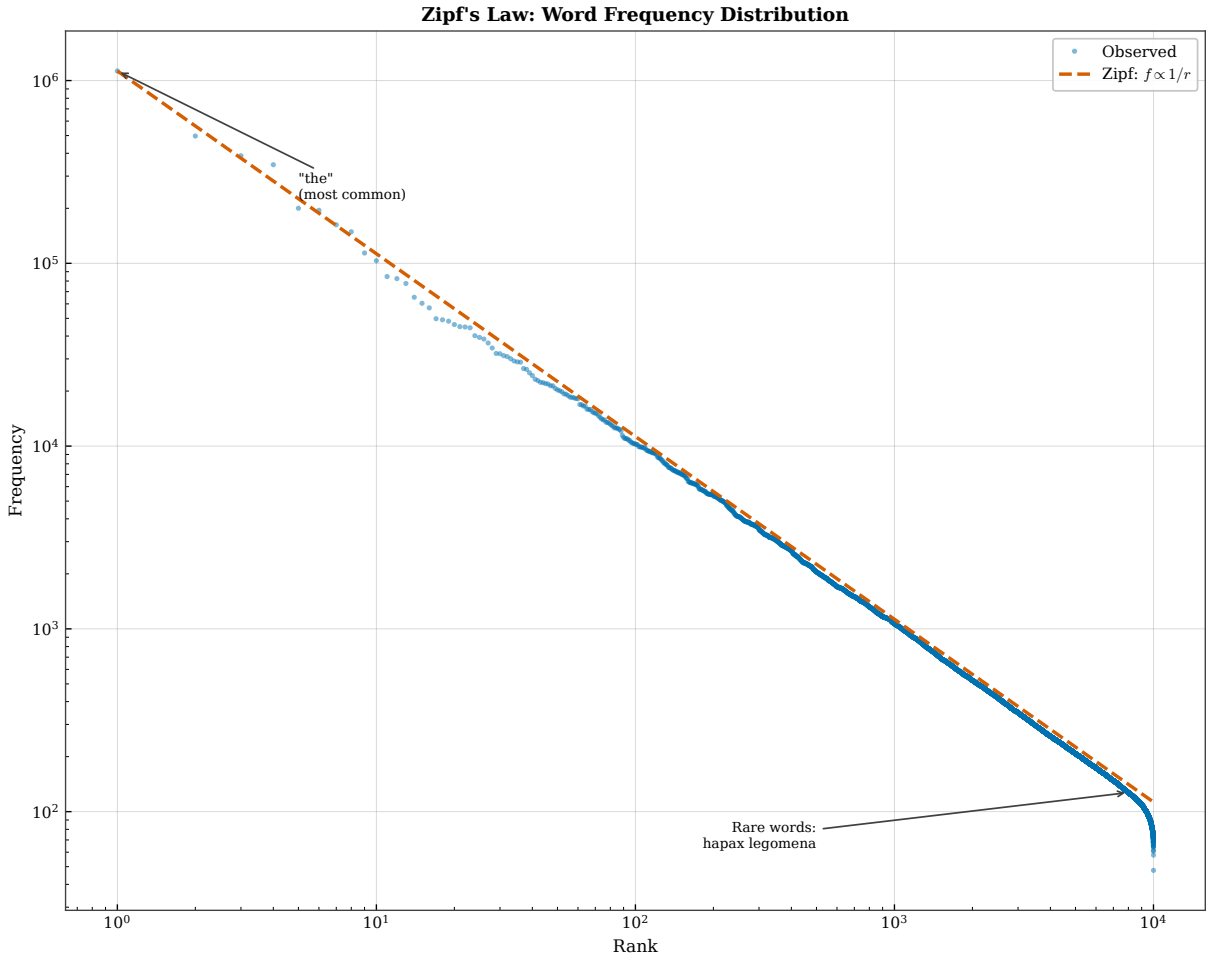

Figure 2.11: Zipf's law and its implications for n-gram sparsity. (a) Word frequencies follow a power law: the most common words are vastly more frequent than the rest. (b) Cumulative coverage shows that a small number of high-frequency words account for most tokens, but the long tail of rare words is essential for complete coverage.

2.4 Smoothing Techniques

Smoothing refers to any technique that assigns non-zero probability to events not observed in training data by redistributing probability mass from observed events. The fundamental principle is simple: we cannot take probability from nowhere, so to give probability to unseen n-grams, we must take some away from seen n-grams. Different smoothing methods embody different philosophies about how much to take, from whom to take it, and how to distribute what is collected. The simplest methods apply uniform adjustments to all n-grams regardless of their frequency, while sophisticated methods tailor their adjustments based on the statistical properties of the training corpus. The development of smoothing techniques spans several decades and represents some of the most elegant applied statistics in natural language processing. In this section, we survey the major approaches, building from the intuitive but flawed Laplace smoothing through the highly effective Kneser-Ney method that remains the gold standard for n-gram language models.

Laplace smoothing, also called add-one smoothing, is the oldest and simplest approach: add one to every count before computing probabilities. If we observe counts $c(w_1, w_2)$ for bigrams, the Laplace-smoothed probability becomes $P_{\text{Laplace}}(w_2|w_1) = \frac{c(w_1,w_2)+1}{c(w_1)+|\mathcal{V}|}$, where the denominator adds the vocabulary size $|\mathcal{V}|$ to maintain proper normalization. Figure 2.12 illustrates this process: a word that appeared 45 times now has count 46, while a never-seen word receives count 1. The formula guarantees that every n-gram receives non-zero probability, solving the immediate problem of zero probabilities. However, Laplace smoothing suffers from

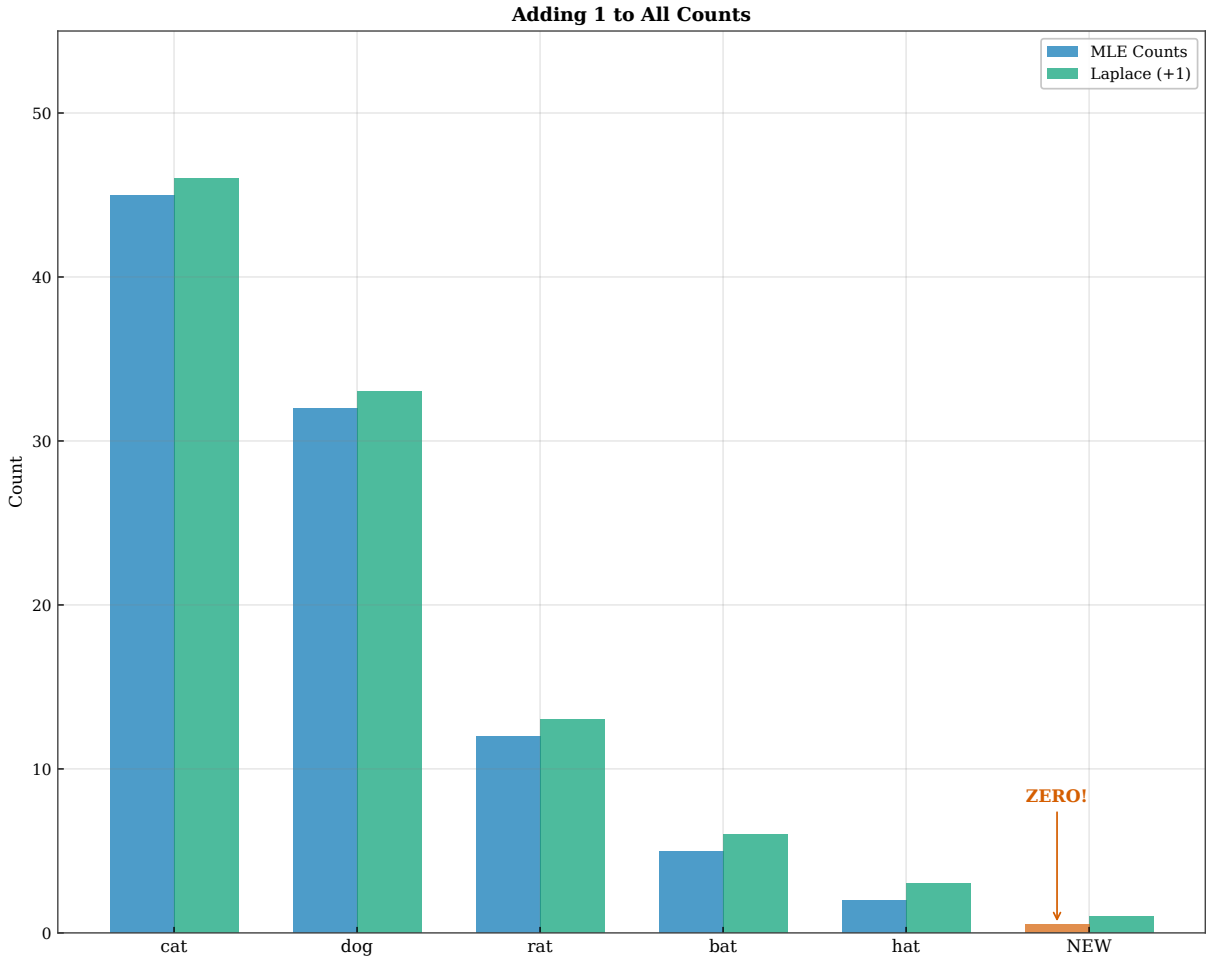

Figure 2.12: Laplace (add-one) smoothing. The left panel shows how adding one to all counts ensures no zero values remain. The right panel compares MLE and Laplace probabilities, demonstrating how previously unseen events (marked "NEW") receive non-zero probability while observed events are slightly discounted.

a severe flaw: it redistributes far too much probability mass to unseen events. With a vocabulary of 50,000 words, adding one to each of 50,000 bigrams following a given word dramatically inflates the denominator, shrinking the probabilities of observed bigrams to near insignificance. The method works adequately for small vocabularies but fails catastrophically for realistic language modeling where vocabularies are large and most n-grams remain unseen.

Add-k smoothing generalizes Laplace by adding a fractional count $k < 1$ instead of one: $P_{\text{add-k}}(w_2|w_1) = \frac{c(w_1,w_2)+k}{c(w_1)+k\cdot|\mathcal{V}|}$. Figure 2.13 shows how different values of k affect the probability distribution. Small values like $k = 0.01$ preserve more of the original distribution's shape, taking less from observed events and giving less to unseen ones. The right panel reveals that optimal k depends on corpus size: with abundant data, less smoothing is needed because MLE estimates are already reliable; with sparse data, more aggressive smoothing prevents overfitting to noise. In practice, k is tuned on held-out data to minimize perplexity, treating it as a hyperparameter rather than a principled choice. While add-k smoothing improves over Laplace, it still applies a uniform adjustment regardless of how often an n-gram appeared. A bigram seen 1,000 times is discounted by the same amount as one seen twice, which seems wasteful—surely we should discount less from events we have strong evidence for and more from events with shaky evidence.

Good-Turing estimation, developed by Alan Turing during World War II for cryptanalysis, offers a more principled approach based on the statistics of the training corpus itself. The key insight is to use the frequency of frequencies: let N_r denote the number of n-grams that appear exactly r times in training. Figure 2.14 shows a typical N_r distribution, which follows a steep power law—many n-grams appear once or twice, few appear

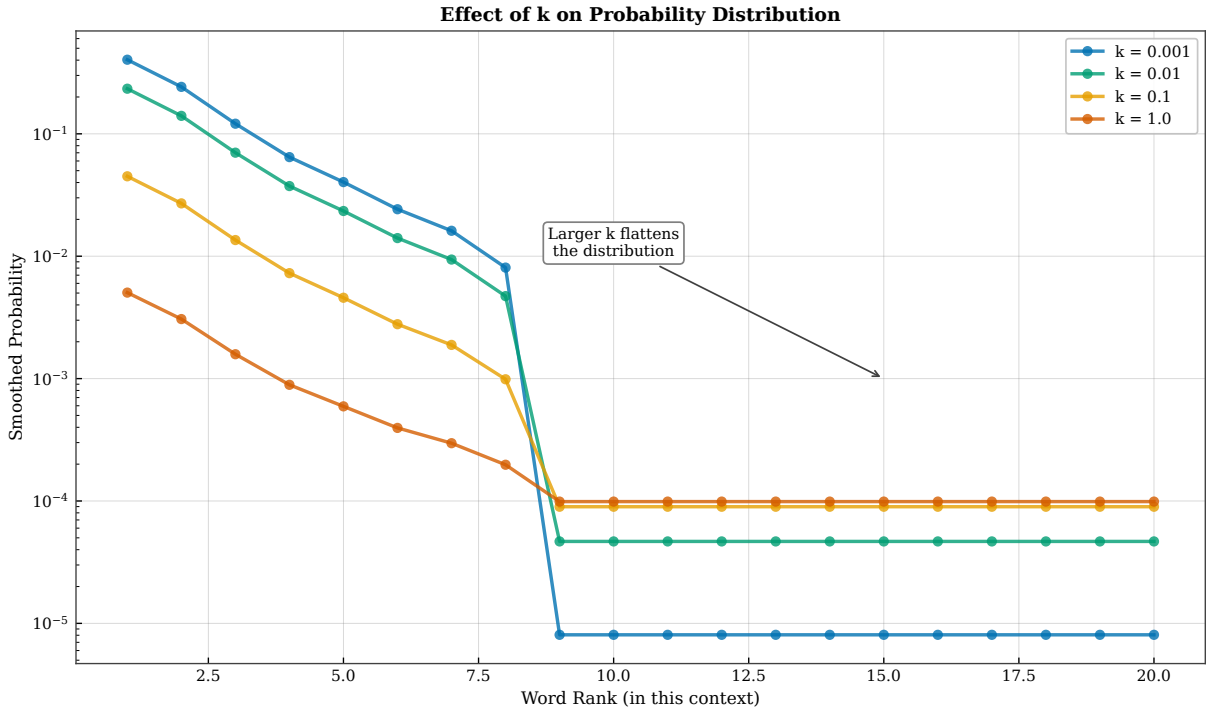

Figure 2.13: Add-k smoothing with different values of k. (a) Larger k values flatten the probability distribution, reducing the gap between frequent and rare words. (b) The optimal k value varies with corpus size: smaller corpora benefit from more smoothing, while larger corpora require less intervention.

thousands of times. Good-Turing proposes adjusting counts according to $c^* = (c+1) \cdot N_{c+1}/N_c$: the adjusted count for items seen c times equals $(c+1)$ multiplied by the ratio of items seen $c+1$ times to items seen c times. This formula has elegant theoretical justification: it estimates what we would expect to observe if we drew a new sample of the same size, accounting for the fact that low-frequency items are more likely to be accidents of sampling than high-frequency items. The total probability mass assigned to unseen events equals N_1/N, where N is the total token count—a quantity entirely determined by how many singletons we observed.

Figure 2.15 visualizes the fundamental trade-off in all smoothing methods: probability is a finite resource that must be divided between observed and unobserved events. MLE dedicates 100% of probability mass to events seen in training, leaving nothing for anything else. Smoothing carves out a portion—say, 15%—for the unseen class, distributing it among all possible n-grams not observed in training. The question every smoothing method must answer is: how much mass should we reserve, and how should we distribute it? Simple methods like add-k use uniform distribution, giving every unseen n-gram equal probability. More sophisticated methods recognize that not all unseen n-grams are equally plausible: "the cat" is more likely than "the the" even if neither appeared in training, because we can infer plausibility from the behavior of individual words in other contexts. This insight leads to the backing-off and interpolation approaches we explore in the next section.

Absolute discounting takes a direct and elegant approach: subtract a fixed amount d (typically around 0.75) from every non-zero count, then redistribute the collected probability mass to unseen events. Figure 2.16 illustrates the mechanism in detail: a bigram with count 100 becomes 99.25, while one with count 5 becomes 4.25, and a singleton with count 1 becomes 0.25. The collected mass is then redistributed to unseen events using a lower-order model, typically a unigram distribution or a smoothed lower-order n-gram. The complete formula becomes $P_{\text{abs}}(w_2|w_1) = \frac{\max(c(w_1,w_2)-d,0)}{c(w_1)} + \lambda(w_1) \cdot P_{\text{lower}}(w_2)$, where $\lambda(w_1)$ is a normalizing factor computed to ensure probabilities sum to one. The intuition behind absolute discounting is that high-count n-grams can afford to lose a fixed amount without much damage to their probability estimates—subtracting 0.75 from a count of 100 changes the estimated probability by less than one percent—while the mass collected from many such small subtractions accumulates into a substantial reserve for unseen events. The optimal discount value d can be derived theoretically using Good-Turing frequency analysis rather than being set arbitrarily: it turns out that

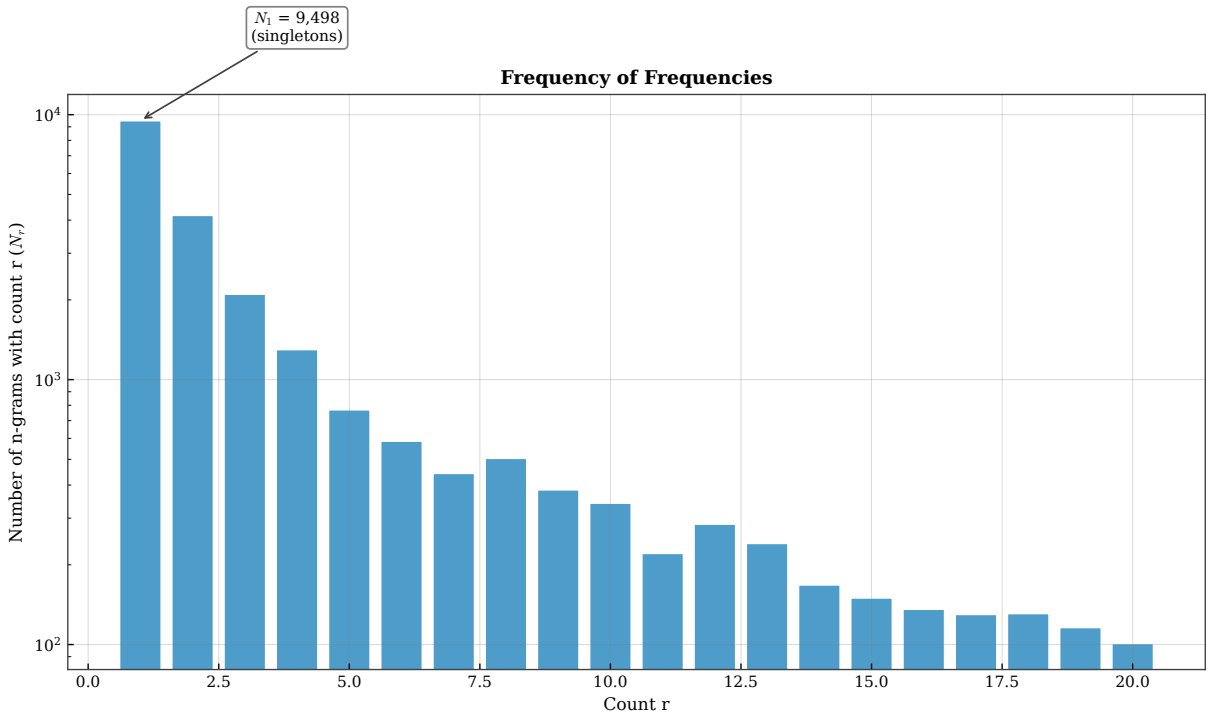

Figure 2.14: Good-Turing estimation. (a) The frequency of frequencies N_r counts how many n-grams appear exactly r times. (b) Good-Turing adjusts counts using the formula $c^* = (c+1) \cdot N_{c+1}/N_c$, systematically reducing observed counts to free mass for unseen events. The shaded region shows mass transferred to unseen n-grams.

$d \approx \frac{N_1}{N_1+2N_2}$, where N_1 and N_2 are the counts of singletons and doubletons respectively, typically yielding values between 0.7 and 0.9 for natural language corpora.

Kneser-Ney smoothing represents the culmination of n-gram smoothing research and remains the best-performing method for statistical language models. Its key innovation addresses a subtle problem with standard backoff: when we back off from an unseen trigram to a bigram, we should not use the raw frequency of the continuation word, because raw frequency conflates versatile words with specialized ones. Figure 2.17 illustrates with a famous example: "Francisco" might appear 10,000 times in a corpus, but almost always following "San." Meanwhile, "glasses" might appear only 1,000 times, but after many different words: "wine glasses," "reading glasses," "sun glasses," and so on. If we encounter an unseen context like "reading ___" and back off to unigram probabilities, raw frequency would favor "Francisco" over "glasses"—clearly wrong, since "reading Francisco" makes no sense. Kneser-Ney solves this by using **continuation probability**: instead of counting how often a word appears, we count in how many different contexts it appears. The continuation probability $P_{\text{cont}}(w)$ is proportional to the number of unique words that precede w in training, capturing the word's versatility rather than its frequency.

The full Kneser-Ney formula combines absolute discounting with continuation probability in an elegant recursive structure: for the highest-order n-gram, we use discounted MLE based on actual counts, and for backoff distributions, we use continuation counts that capture word versatility rather than raw frequency. For bigrams, the formula becomes: $P_{\text{KN}}(w_2|w_1) = \frac{\max(c(w_1,w_2)-d,0)}{c(w_1)} + \lambda(w_1) \cdot P_{\text{cont}}(w_2)$, where the continuation probability $P_{\text{cont}}(w_2) = \frac{|\{v:c(v,w_2)>0\}|}{|\{(v,w):c(v,w)>0\}|}$ counts the number of unique predecessor words that appear before w_2 in the training data, divided by the total number of unique bigram types observed. This elegant formulation captures the profound intuition that words appearing in diverse contexts are more likely to be appropriate continuations for novel contexts than words locked into specific fixed phrases, regardless of how frequently those fixed phrases occur. Modified Kneser-Ney, developed by Chen and Goodman in the late 1990s, further improves performance by using three different discount values d_1, d_2, and d_{3+} for n-grams appearing exactly once, exactly twice, and

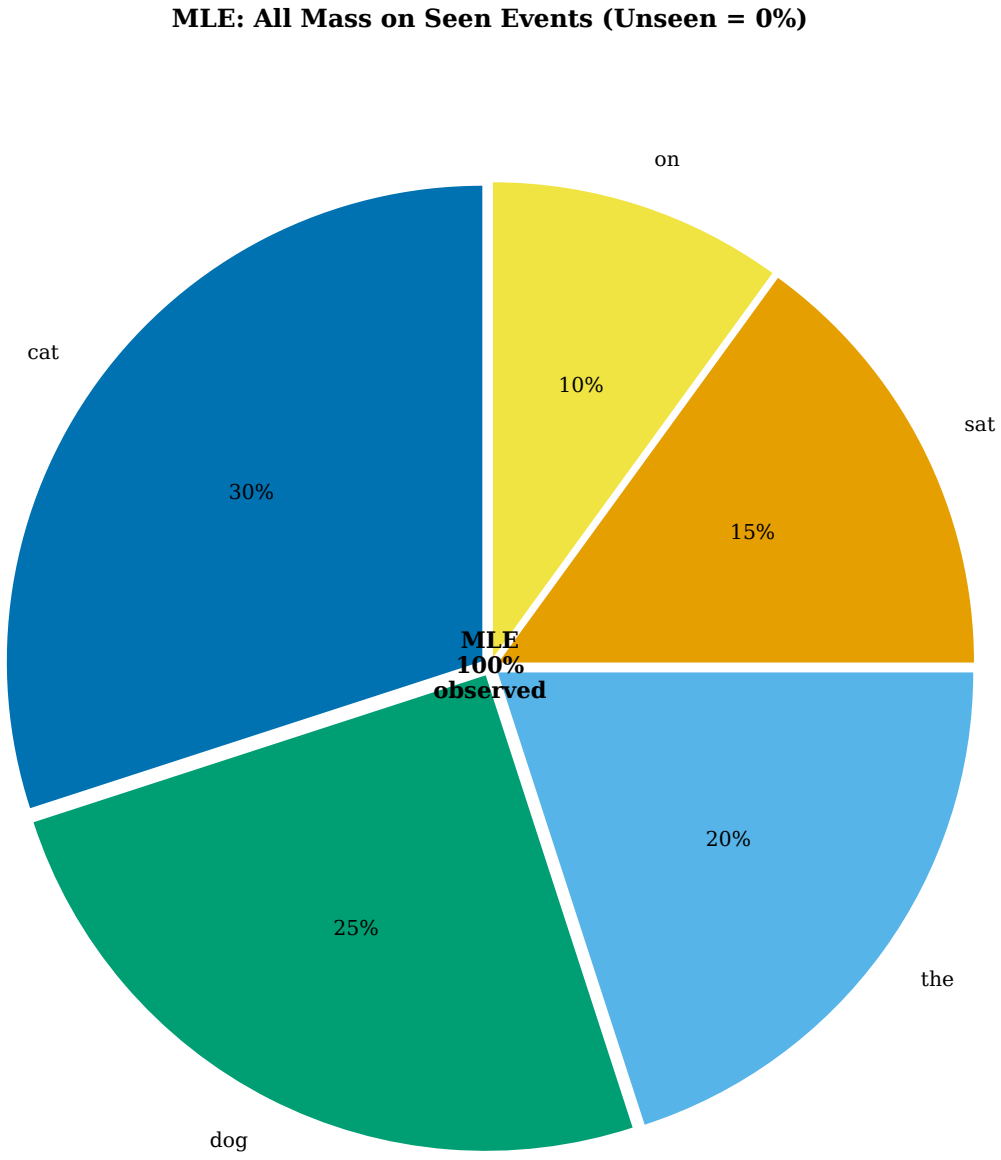

Figure 2.15: Probability mass redistribution in smoothing. MLE assigns 100% of probability mass to observed events, leaving nothing for unseen words. Smoothing reserves a portion (here 15%) for the unseen class, enabling the model to handle novel inputs while still preferring observed patterns.

three or more times respectively, recognizing that the optimal discount differs based on how much evidence we have for each n-gram. The method extends naturally to higher-order n-grams by recursively applying the same continuation probability principle at each backoff level, creating a hierarchy where each level contributes its unique statistical evidence to the final probability estimate.

Figure 2.18 compares smoothing methods empirically on standard benchmark datasets, revealing the substantial practical differences between approaches. The left panel shows perplexity scores for various methods on held-out test data: no smoothing gives infinite perplexity because any single zero probability in the chain rule product causes the entire computation to fail, Laplace smoothing produces poor results by redistributing too much mass uniformly, add-k improves substantially by allowing tuning of the smoothing strength, Good-Turing performs well by using corpus statistics to guide redistribution, and Kneser-Ney achieves the best performance

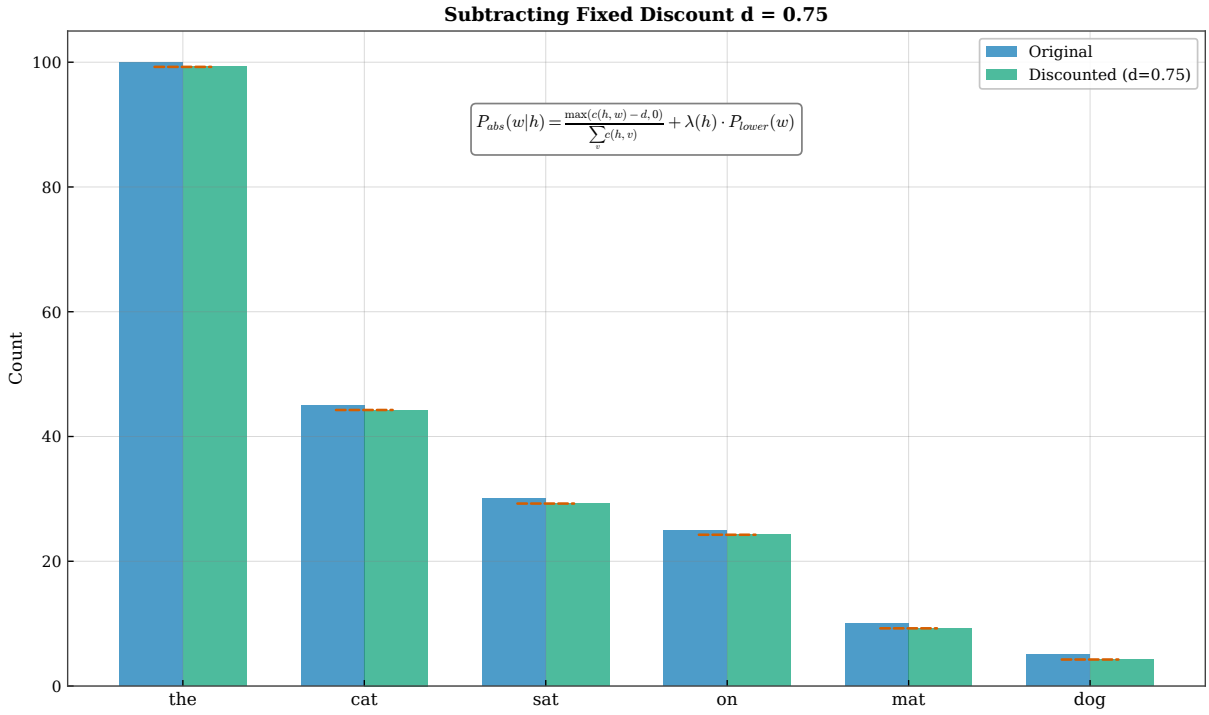

Figure 2.16: Absolute discounting. (a) A fixed discount d (typically 0.75) is subtracted from each non-zero count. (b) The total mass collected depends on d and the count distribution, with higher d values extracting more mass for redistribution to unseen events.

by incorporating word versatility information. The right panel reveals how different methods scale with n-gram order: simple methods like Laplace deteriorate rapidly as n increases because they cannot cope with the exponentially growing sparsity that characterizes higher-order models, while Kneser-Ney maintains reasonable and relatively stable performance even for 5-grams by effectively sharing statistical strength across related contexts. This robustness explains why Kneser-Ney became the standard choice for n-gram language models in demanding applications like speech recognition and machine translation, where higher-order models are essential to capture phrase-level patterns and idiomatic expressions. The development from Laplace through Good-Turing to Kneser-Ney represents an intellectual journey from naive uniformity to sophisticated statistical inference, progressively learning to use the inherent structure of natural language to inform how probability mass should be distributed among unseen events.

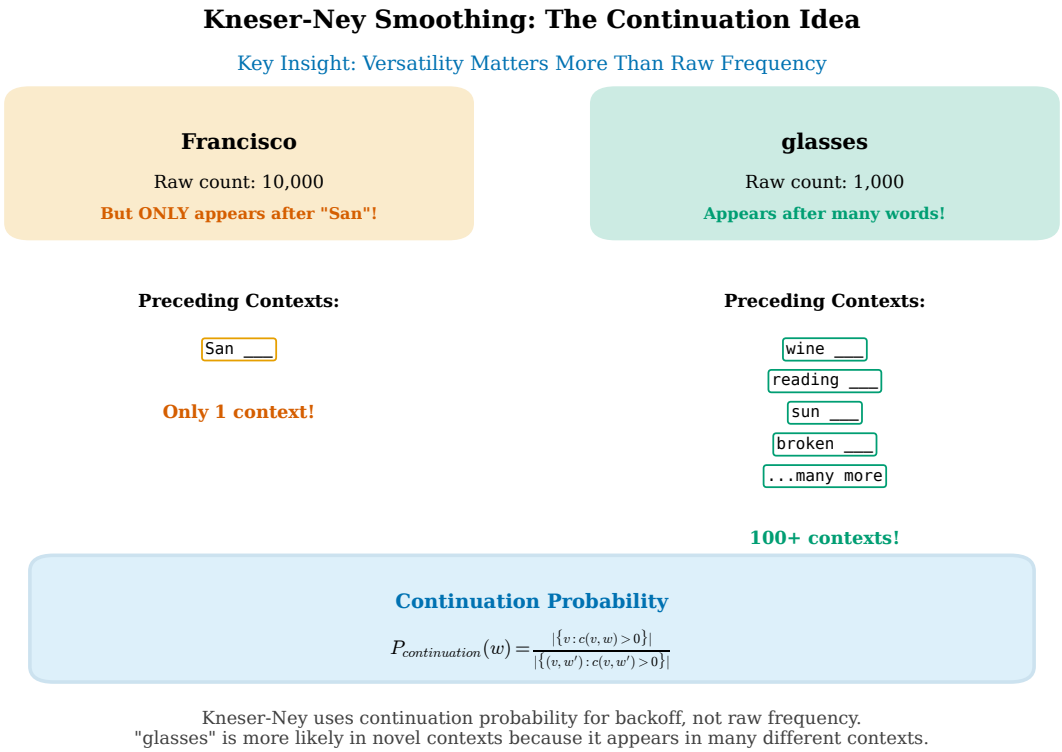

Figure 2.17: The key insight of Kneser-Ney smoothing: continuation probability. “Francisco” has high raw frequency but appears only after “San,” making it a poor choice for novel contexts. “glasses” has lower frequency but appears after many different words, making it more versatile. Kneser-Ney uses continuation probability rather than raw frequency for backing off.

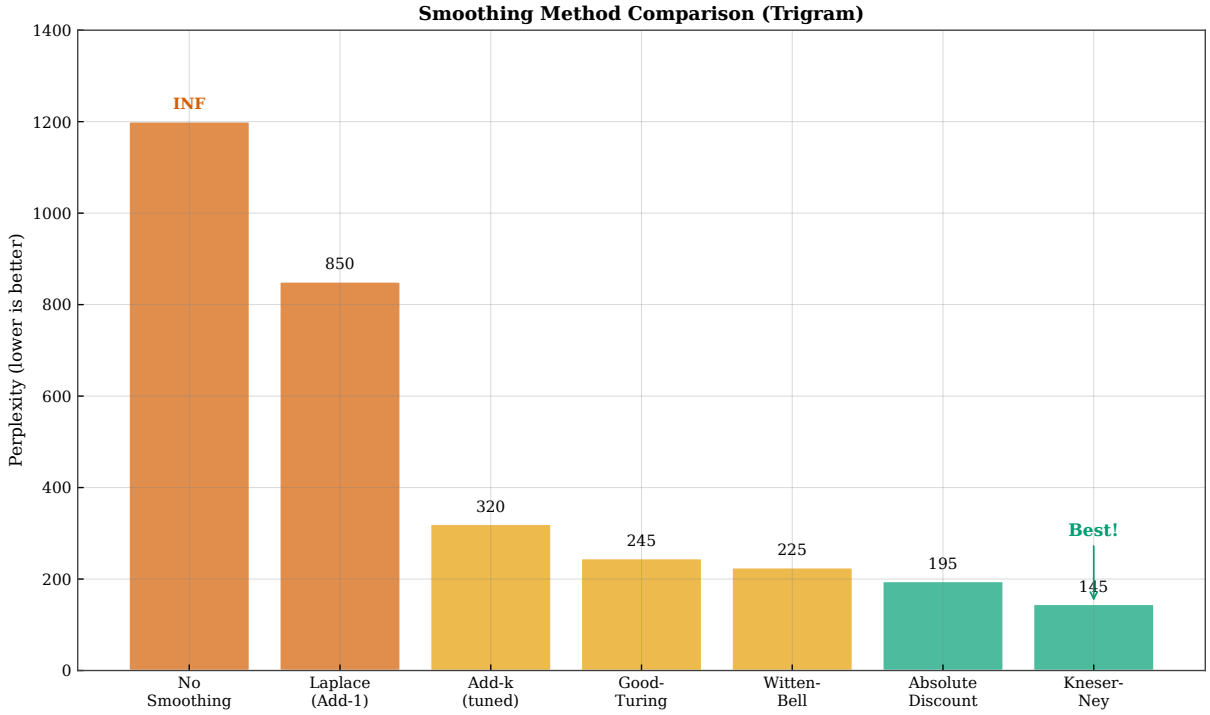

Figure 2.18: Comparison of smoothing methods. (a) Perplexity comparison for trigram models: Kneser-Ney achieves the lowest perplexity, while no smoothing yields infinite perplexity due to zero probabilities. (b) Performance across n-gram orders: Kneser-Ney maintains low perplexity even for higher-order models where other methods deteriorate.

2.5 Backoff and Interpolation

Smoothing techniques address the zero-probability problem by ensuring that no n-gram receives exactly zero probability, but they leave open an important architectural question: how should we combine information from different n-gram orders when making predictions? A trigram model conditions on two preceding words, capturing phrase-level patterns, but what if that specific two-word context is rare or completely unseen in the training corpus? We could use the trigram probability if sufficient evidence is available, but fall back to more reliable bigram or unigram probabilities when the trigram evidence is sparse or absent. This intuition leads to two complementary strategies for combining n-gram orders: **backoff**, which uses higher-order estimates when possible and falls back to lower orders only when direct evidence is lacking, and **interpolation**, which always combines estimates from all available orders using learned weights that reflect each order's reliability. Both strategies exploit a crucial insight about the bias-variance trade-off in statistical estimation: lower-order models have worse predictive accuracy because they ignore relevant context but enjoy better coverage because their simpler patterns appear more frequently, while higher-order models have better potential accuracy because they consider more context but suffer more severely from sparsity because their complex patterns rarely repeat. The art of n-gram modeling lies in balancing these competing trade-offs to achieve the best overall performance on held-out data.

Figure 2.19 illustrates the backoff strategy with a concrete example. To estimate $P(\text{cushion}|\text{velvet})$, we first check whether the bigram "velvet cushion" appeared in training. If the count is positive, we use the (possibly discounted) MLE estimate. If the count is zero, we back off to the unigram probability $P(\text{cushion})$, multiplied by a normalizing factor $\alpha(\text{velvet})$ that ensures the conditional distribution sums to one. The factor α is computed to redistribute exactly the probability mass that was discounted from observed bigrams, maintaining proper normalization. For trigram models, we might back off twice: first from the unseen trigram to a bigram, then potentially from an unseen bigram to a unigram. The hierarchy continues for higher-order models, always falling back to more general estimates when specific evidence is lacking. The key principle is that we only back off when forced to by zero counts; when we have evidence at the higher order, we trust it.

Interpolation takes a fundamentally different approach: rather than choosing between n-gram orders based on whether counts are zero or positive, we always combine estimates from all available orders using learned weights, creating a weighted average that benefits from each order's strengths. Figure 2.20 shows the linear interpolation formula: $P_{\text{interp}}(w|\text{context}) = \lambda_3 \cdot P_3(w|w_{-2}, w_{-1}) + \lambda_2 \cdot P_2(w|w_{-1}) + \lambda_1 \cdot P_1(w)$, where the interpolation weights λ_i are non-negative and sum to one, ensuring the result is a valid probability distribution. The intuition is that even when we have trigram evidence from direct observation, the bigram and unigram estimates provide useful complementary information that can regularize our predictions, especially when the trigram count is low and therefore unreliable as a probability estimate. The weights λ_i can be global constants learned on held-out development data to minimize perplexity, or they can vary dynamically by context, with rare contexts receiving higher weights on lower-order models whose estimates are more reliable and frequent contexts trusting higher-order estimates that have sufficient data support. Context-dependent interpolation, shown in the right panel of the figure, adapts the weighting to the statistical reliability of each specific context: a trigram context observed 1,000 times warrants high trigram weight because we have strong evidence, while a context seen only twice should rely more heavily on the robust lower-order estimates that draw from much larger pools of evidence.

When corpora become truly massive—billions or trillions of words from web crawls—the sophisticated machinery of Kneser-Ney smoothing becomes less necessary because raw counts are more reliable. **Stupid backoff**, introduced by researchers at Google, exploits this observation with a radically simple approach: use relative frequency when the n-gram count is positive, otherwise multiply by a fixed factor $\alpha = 0.4$ and back off to the next lower order. Figure 2.21 shows the algorithm and its performance characteristics. The method produces "scores" rather than probabilities because the factors do not normalize properly, but for applications like ranking candidate translations or speech recognition hypotheses, relative scores suffice. The right panel reveals a striking finding: at web scale (billions of tokens), stupid backoff matches or exceeds Kneser-Ney performance despite its simplicity. This happens because with enough data, even 5-grams become reasonably well-estimated, reducing the need for careful probability redistribution. The lesson is that the optimal complexity of a smoothing method depends on the data regime: sophisticated methods shine with limited data, while

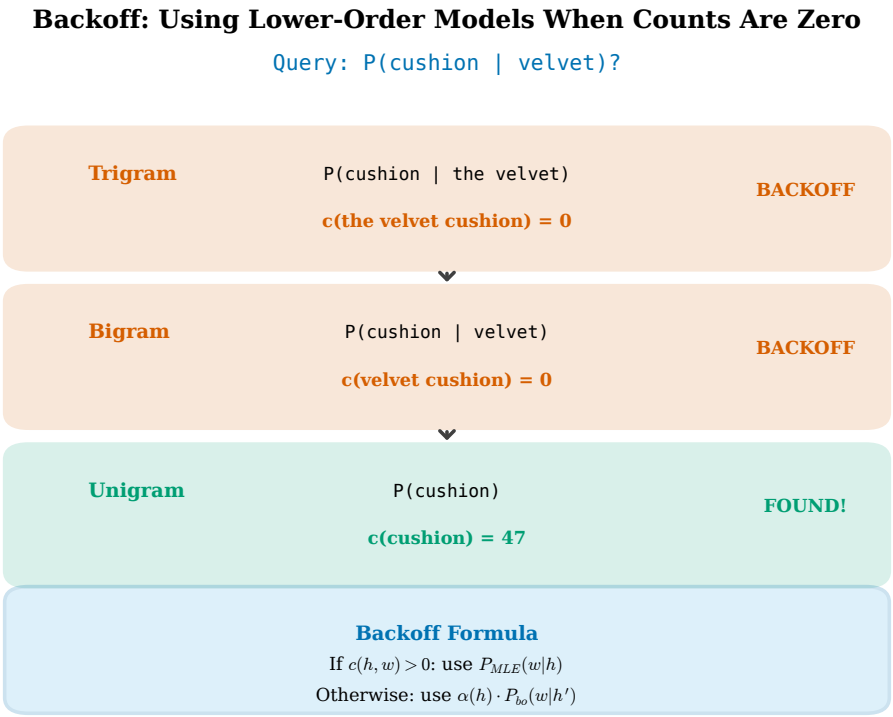

Figure 2.19: The backoff strategy for handling unseen n-grams. When estimating $P(\text{cushion}|\text{velvet})$, we first check if the bigram was observed. If the count is zero, we back off to the unigram probability, applying a normalization factor α to maintain a valid probability distribution.

simple methods suffice when data is abundant.

Figure 2.22 provides a three-dimensional view of the backoff hierarchy, visualizing the fundamental trade-off between specificity and coverage. At the top, 5-gram models make highly specific predictions conditioned on four preceding words, but the pyramid narrows because only a tiny fraction of possible 5-grams appear in any corpus. Moving down, each level broadens: 4-grams, trigrams, bigrams, and finally unigrams, which cover the entire vocabulary but ignore context entirely. The arrows show the backoff direction: when a higher-order count is zero, we descend to more general estimates. This hierarchy embodies a deep principle: we should use the most specific model for which we have reliable evidence, falling back to generality only when specificity fails. The same principle reappears in neural language models, where attention mechanisms learn to focus on relevant context while ignoring irrelevant history, and in hierarchical architectures where lower layers capture local patterns and higher layers integrate global context.

The choice between backoff and interpolation, and the specific smoothing method used at each level of the n-gram hierarchy, depends on the application requirements and available computational resources. For real-time applications like speech recognition and predictive text input, computational efficiency matters greatly, favoring simpler methods that can be implemented with efficient data structures like tries, minimal perfect hash functions, and compressed suffix arrays that enable constant-time lookups. For offline applications like corpus analysis, machine translation training, or research benchmarking, more sophisticated methods with higher computational overhead can be afforded because latency is less critical than accuracy. Modified Kneser-Ney with interpolation remains the gold standard for moderate-sized corpora in the millions to billions of tokens, providing the best balance of accuracy and reasonable computational requirements. Stupid backoff dominates at web scale with trillions of tokens, where raw data abundance compensates for algorithmic simplicity. The common thread across all these methods is the recognition that language modeling is fundamentally a problem of trading off specificity against reliability: we want to use the most specific context available while gracefully

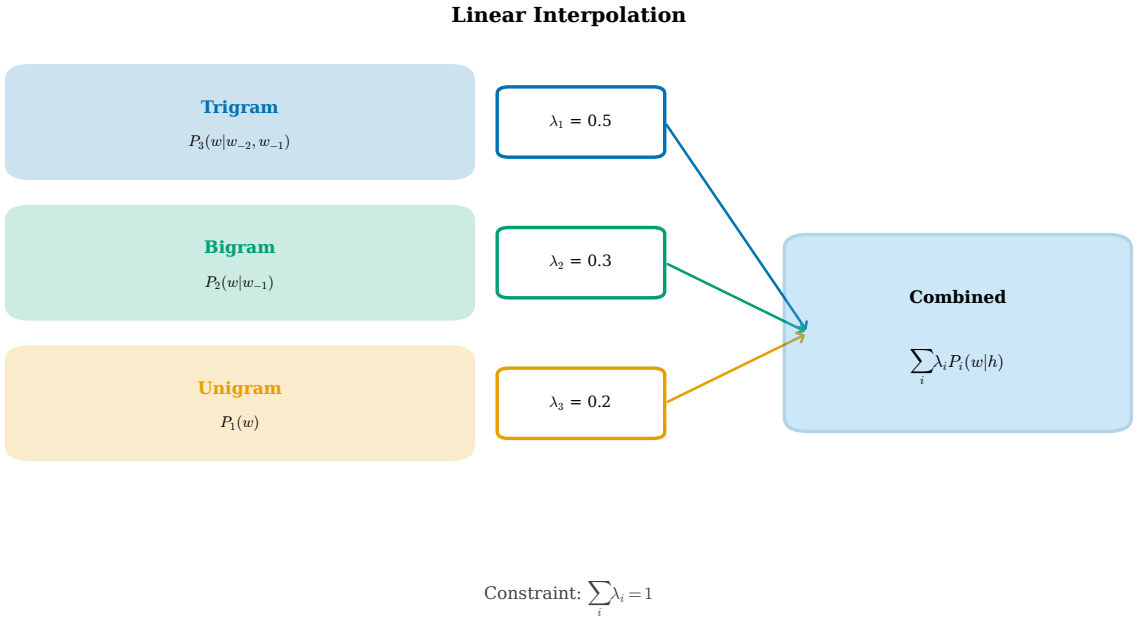

Figure 2.20: Interpolation combines estimates from all n-gram orders using learned weights. (a) The linear interpolation formula weights trigram, bigram, and unigram probabilities. (b) Context-dependent weights allow the model to rely more on lower-order estimates for rare contexts where higher-order statistics are unreliable.

degrading to more general estimates when specific evidence is sparse or absent. This fundamental trade-off, first encountered and systematically studied in n-gram models, will recur throughout our exploration of neural language models in subsequent chapters, where analogous decisions about model capacity, regularization strength, and effective context length reflect the same underlying tension between fitting the training data and generalizing to new inputs.

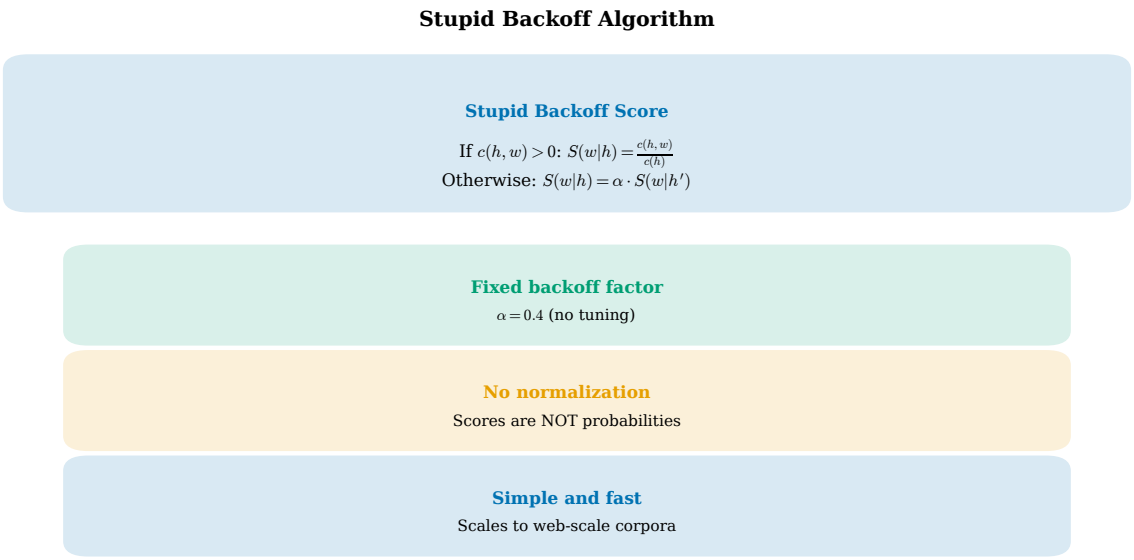

Figure 2.21: Stupid backoff: a simple method that scales to web-size corpora. The algorithm uses a fixed backoff factor $\alpha = 0.4$ without normalization, producing scores rather than probabilities. At massive scale, stupid backoff matches the performance of more sophisticated methods like Kneser-Ney while being simpler and faster.

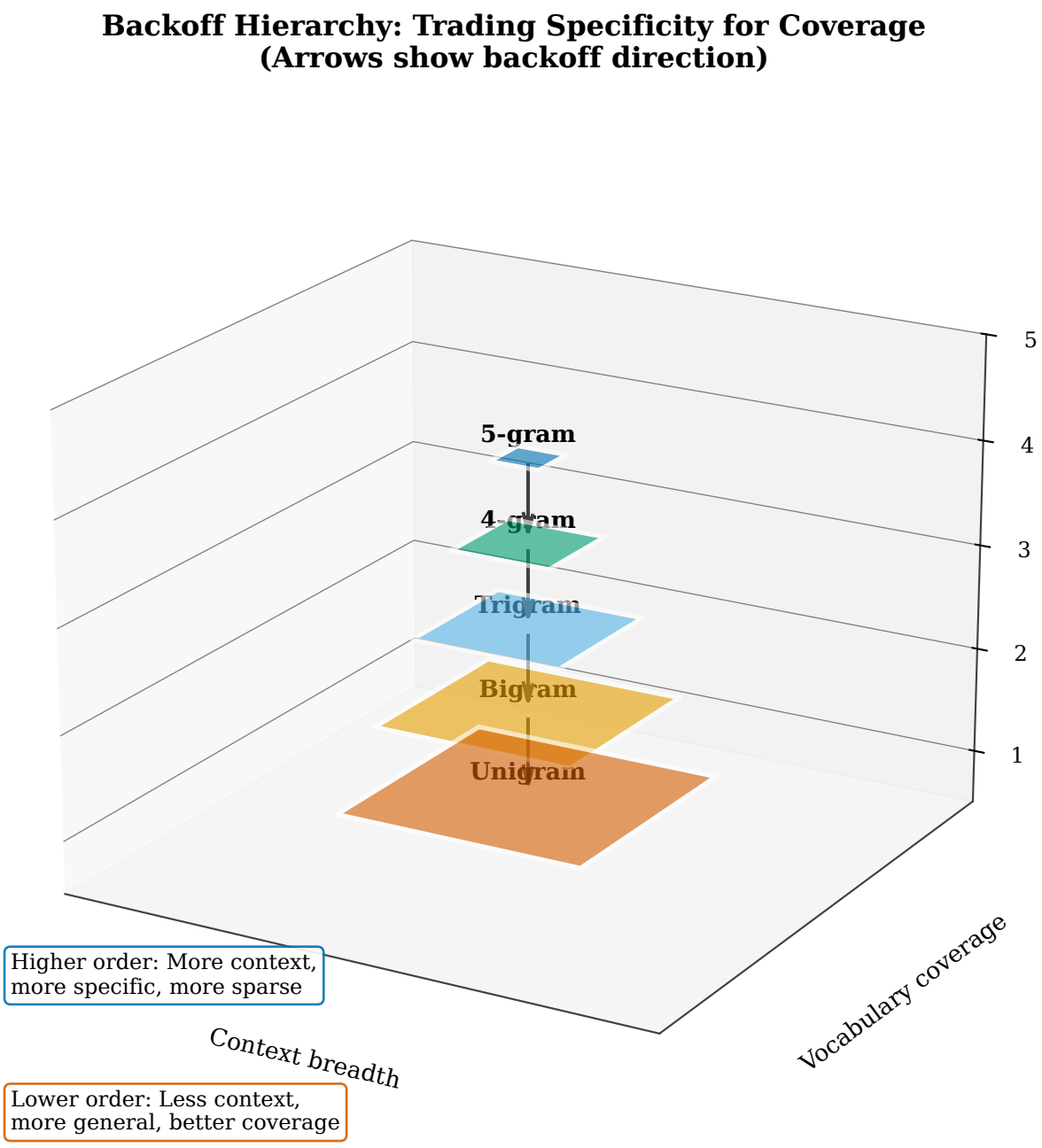

Figure 2.22: 3D visualization of the backoff hierarchy. Higher-order n-grams (top) provide more specific predictions but cover fewer contexts. Lower-order n-grams (bottom) cover more contexts but make less specific predictions. Arrows indicate the backoff direction when higher-order counts are zero.

2.6 Evaluation and Limitations

Having developed the complete machinery of n-gram language models—from the Markov assumption that makes estimation tractable, through maximum likelihood estimation that converts counts to probabilities, smoothing techniques that handle the sparsity problem, and backoff strategies that combine evidence from multiple n-gram orders—we now turn to the critical questions of evaluating model quality and understanding the fundamental limitations of this approach. The standard metric for language model evaluation is **perplexity**, an information-theoretic measure that quantifies how well a probability model predicts a held-out test set that was not used during training. Lower perplexity indicates a better model: a model that assigns higher probability to the test data is less "perplexed" or surprised by what it sees, suggesting it has learned the underlying statistical patterns of the language. Understanding perplexity deeply requires connecting language modeling to Shannon's foundational information theory, revealing that evaluating language models is fundamentally about measuring how many bits are needed to encode text under the model's probability distribution—a model that predicts well requires fewer bits because it anticipates what comes next. Beyond establishing evaluation metrics, we must also confront the inherent limitations of n-gram models that no amount of clever engineering can overcome, limitations that ultimately motivate the neural approaches developed in subsequent chapters.

Perplexity is defined mathematically as the inverse probability of the test set, normalized by the number of words to enable comparison across texts of different lengths: $PP(W) = P(w_1, w_2, \ldots, w_N)^{-1/N}$. Figure 2.23 illustrates this formula and provides intuition for its interpretation. A perplexity of 100 means the model is, on average, as uncertain about the next word as if it were choosing uniformly among 100 equally likely candidates—the model's probability distribution has an effective support size of 100 words at each position. The geometric mean formulation ensures that perplexity reflects average uncertainty per word rather than total uncertainty over the entire test set, making the metric comparable across corpora of different sizes. This normalization also means that perplexity is multiplicative across independent choices: if we are equally uncertain at each position, the per-word perplexity captures this regardless of sequence length. For comparison across the history of language modeling: a random model choosing uniformly from a 50,000-word vocabulary has perplexity 50,000 by definition; a unigram model that predicts based on word frequency alone might achieve perplexity around 1,000 by learning that common words are more likely; a well-tuned trigram model reaches 100-200 by capturing local sequential dependencies; and state-of-the-art neural models achieve perplexities below 50 on standard benchmarks by learning long-range patterns and semantic relationships. The right panel of the figure shows this progression, illustrating how each advance in modeling sophistication yields measurable and substantial improvement in perplexity.

Perplexity connects directly to **cross-entropy**, a fundamental concept in information theory. The cross-entropy of a language with respect to a model is $H(L,M) = -\frac{1}{N}\sum_{i=1}^{N} \log_2 P_M(w_i|w_{<i})$, measuring the average number of bits needed to encode each word under the model's distribution. Figure 2.24 shows that perplexity is simply $2^{H(L,M)}$: if a model has cross-entropy of 7 bits per word, its perplexity is $2^7 = 128$. The information-theoretic perspective reveals a profound insight: cross-entropy provides an upper bound on the true entropy of language. If there existed a perfect model that captured all statistical regularities of language, its cross-entropy would equal the true entropy—the minimum number of bits needed to encode text. Every improvement in language modeling reduces cross-entropy toward this theoretical limit, with the gap indicating how much predictable structure the model fails to capture. Shannon himself estimated English entropy at roughly 1 bit per character through human guessing experiments, suggesting that English text is highly redundant and that significant room remains for improvement even in modern language models.

Despite sophisticated smoothing and the success of n-gram models in practical applications, they suffer from fundamental limitations that no amount of engineering can overcome. Figure 2.25 summarizes these limitations. First, the **bounded context window** means n-grams cannot capture dependencies beyond $n-1$ words. Consider "The trophy would not fit in the suitcase because it was too [large/small]"—resolving "it" to either "trophy" or "suitcase" requires understanding the entire sentence, not just the preceding few words. Second, n-grams have **no semantic understanding**: "cat" and "feline" are treated as completely unrelated symbols despite being synonyms. A bigram model that learns "the cat sat" cannot transfer this knowledge to predict "the feline sat." Third, the **exponential parameter growth**—$|\mathcal{V}|^n$ possible n-grams—makes high-order models impractical for realistic vocabularies. Fourth, n-grams cannot **generalize** across similar contexts:

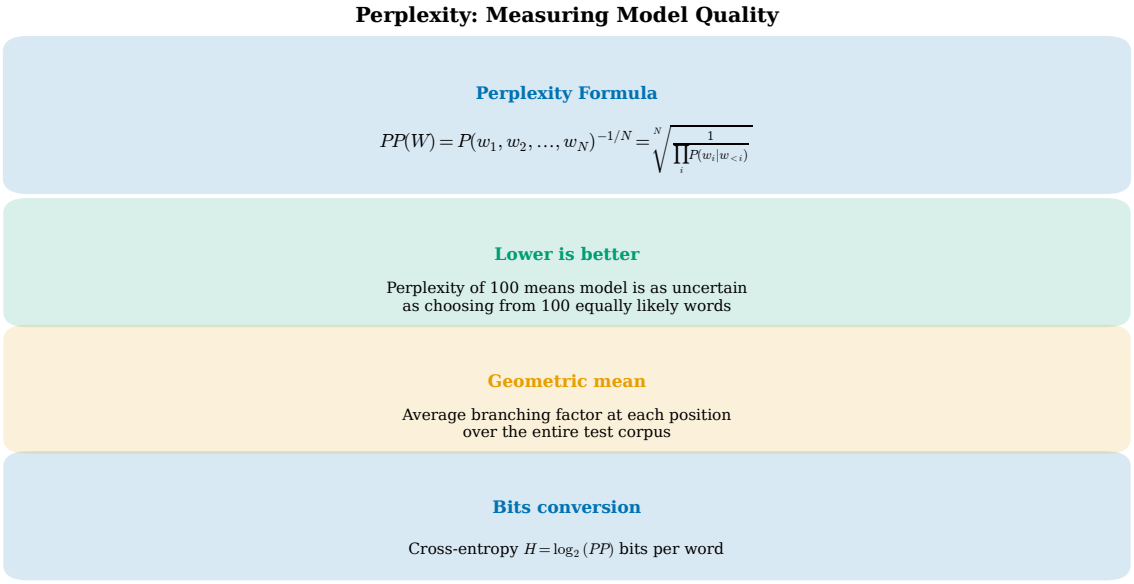

Figure 2.23: Perplexity as a measure of model quality. The formula computes the geometric mean of inverse probabilities across all test words. Lower perplexity indicates better prediction: a perplexity of 100 means the model is as uncertain as choosing uniformly from 100 words. The right panel compares models, from random baseline through neural language models.

seeing “the red car” provides no information about “the crimson automobile” because each n-gram is learned independently without recognizing the underlying semantic similarity.

Figure 2.26 shows concrete next-word prediction examples that illustrate both the strengths and weaknesses of n-gram models. For common patterns like “The cat sat on the ___,” a well-trained trigram model correctly assigns high probability to “mat,” “floor,” and other plausible continuations. Similarly, for the famous phrase “The quick brown ___,” the model has memorized “fox” as the likely continuation. However, the model fails for novel entities: after “The president of ___,” it cannot predict a newly relevant organization like “OpenAI” because it has never seen this specific phrase in training. Rare words pose similar challenges: after “She picked up the ___,” common objects like “phone” and “book” receive high probability, but unusual continuations like “gauntlet” are effectively impossible even when contextually appropriate. These failures reveal that n-gram models are fundamentally limited to reproducing patterns they have explicitly observed, with no capacity for the compositional generalization that humans effortlessly achieve.

The limitations of n-gram models directly motivate the neural language models we develop in subsequent chapters, and understanding these limitations illuminates why neural approaches represent such a fundamental advance rather than merely an incremental improvement. Neural networks address the bounded context problem through architectures like recurrent neural networks that maintain hidden state vectors propagating information across arbitrary distances, and Transformers that use attention mechanisms to directly connect any two positions in a sequence regardless of their separation. They overcome the lack of semantic understanding by learning distributed word representations where semantically similar words occupy nearby points in a continuous vector space, enabling transfer of knowledge between synonyms, analogous contexts, and related concepts that would be completely independent in the n-gram framework. They avoid the curse of dimensionality and exponential parameter growth by sharing the same neural network weights across all positions in the sequence, with model size growing linearly with embedding dimension and layer count rather than exponentially with context length. Most importantly, neural models generalize across similar contexts because the same learned transformations process all inputs, identifying abstract patterns like “adjective before noun”

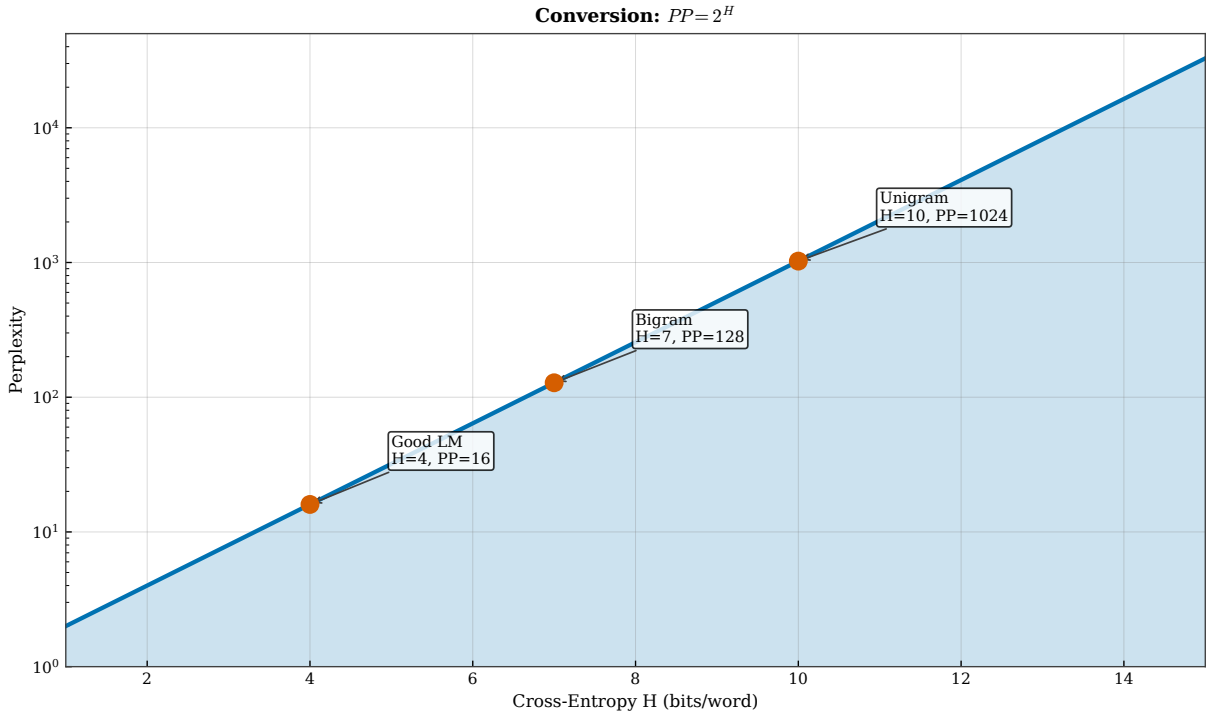

Figure 2.24: The relationship between cross-entropy and perplexity. (a) Perplexity is the exponential of cross-entropy: $PP = 2^H$. (b) Cross-entropy provides an upper bound on the true entropy of language, with better models approaching this theoretical limit.

or "verb requires object" that transfer automatically to novel word combinations never seen during training. Understanding n-gram models and their fundamental limitations is thus essential preparation for appreciating the magnitude of the advance that neural approaches provide, and for understanding why language modeling capabilities improved so dramatically when these architectural innovations became computationally feasible.

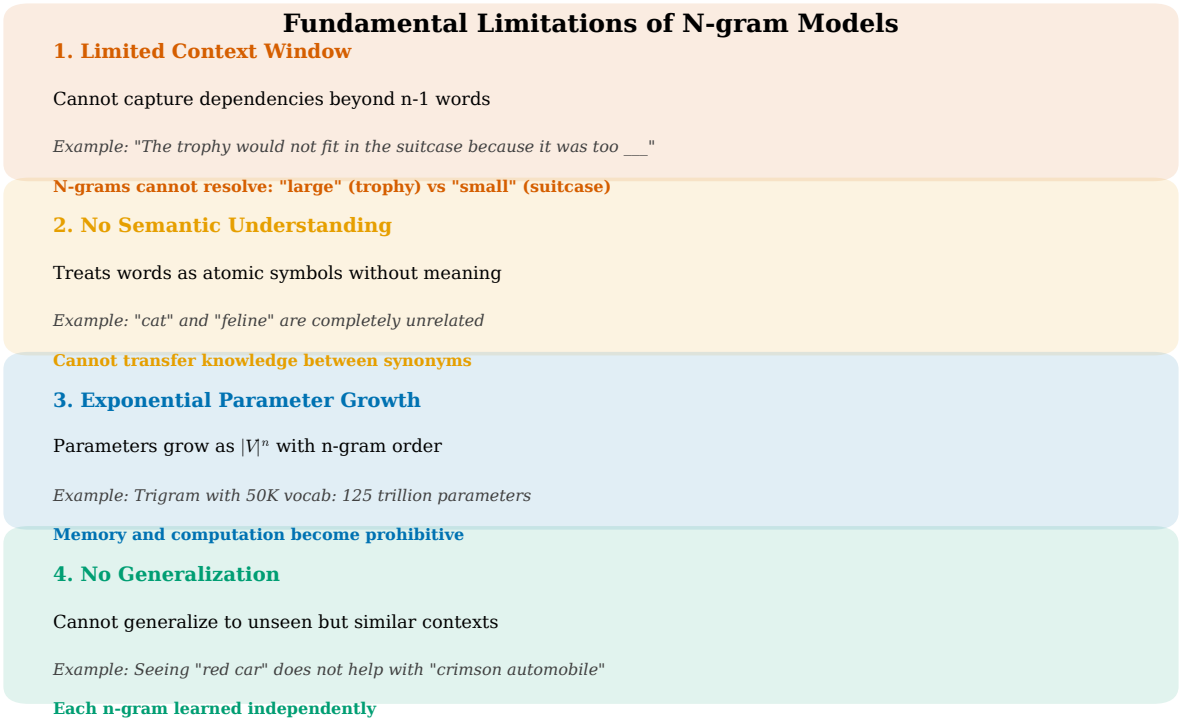

Figure 2.25: Fundamental limitations of n-gram language models. These limitations—bounded context, lack of semantic understanding, exponential parameter growth, and inability to generalize—motivate the development of neural language models that can overcome these barriers.

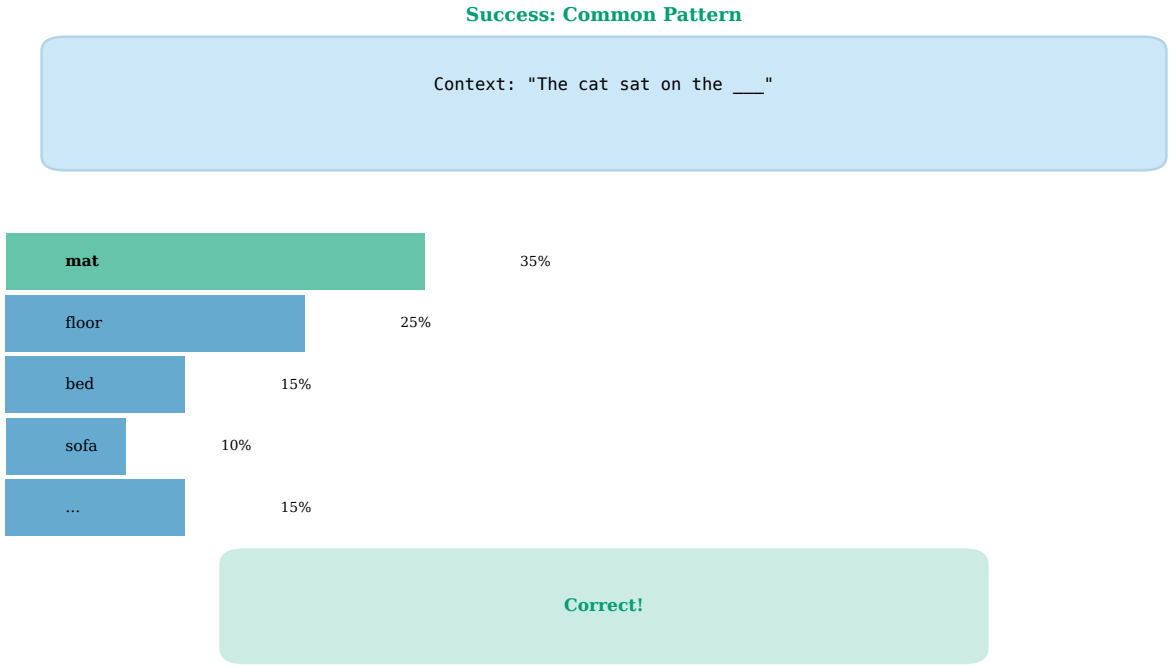

Figure 2.26: Next-word prediction examples showing n-gram successes and failures. Common patterns and memorized phrases are predicted well, but novel entities and rare words expose the model's limitations. These failure cases illustrate why more powerful architectures are needed.

2.7 Summary

This chapter introduced n-gram language models, the foundational approach to next-word prediction that dominated natural language processing for over three decades. Figure 2.27 provides a visual summary of the key concepts. We began with the **Markov assumption**, which makes language modeling tractable by limiting context to the preceding $n-1$ words. This assumption enables **maximum likelihood estimation** through simple count ratios: the probability of a word given its context equals the count of the n-gram divided by the count of the context. However, the **sparsity problem** immediately confronts us: most possible n-grams never appear in training, and raw MLE assigns them zero probability, causing catastrophic failures on test data.

Smoothing techniques address sparsity by redistributing probability mass from observed to unobserved events. We traced the evolution from naive Laplace smoothing through sophisticated Kneser-Ney, which uses continuation probability to capture word versatility rather than raw frequency. **Backoff and interpolation** provide complementary strategies for combining evidence from different n-gram orders, falling back to more reliable lower-order estimates when higher-order evidence is sparse. Finally, we examined **perplexity** as the standard evaluation metric and confronted the **fundamental limitations** of n-gram models: bounded context, lack of semantic understanding, exponential parameter growth, and inability to generalize across similar contexts.

The concepts introduced in this chapter—probability distributions over sequences, the trade-off between context and sparsity, the principle of redistributing probability mass, and the use of perplexity for evaluation—will recur throughout our exploration of neural language models. While neural approaches overcome many n-gram limitations, they face analogous challenges of balancing model capacity against generalization, and the intuitions developed here provide essential foundation for understanding more advanced architectures.

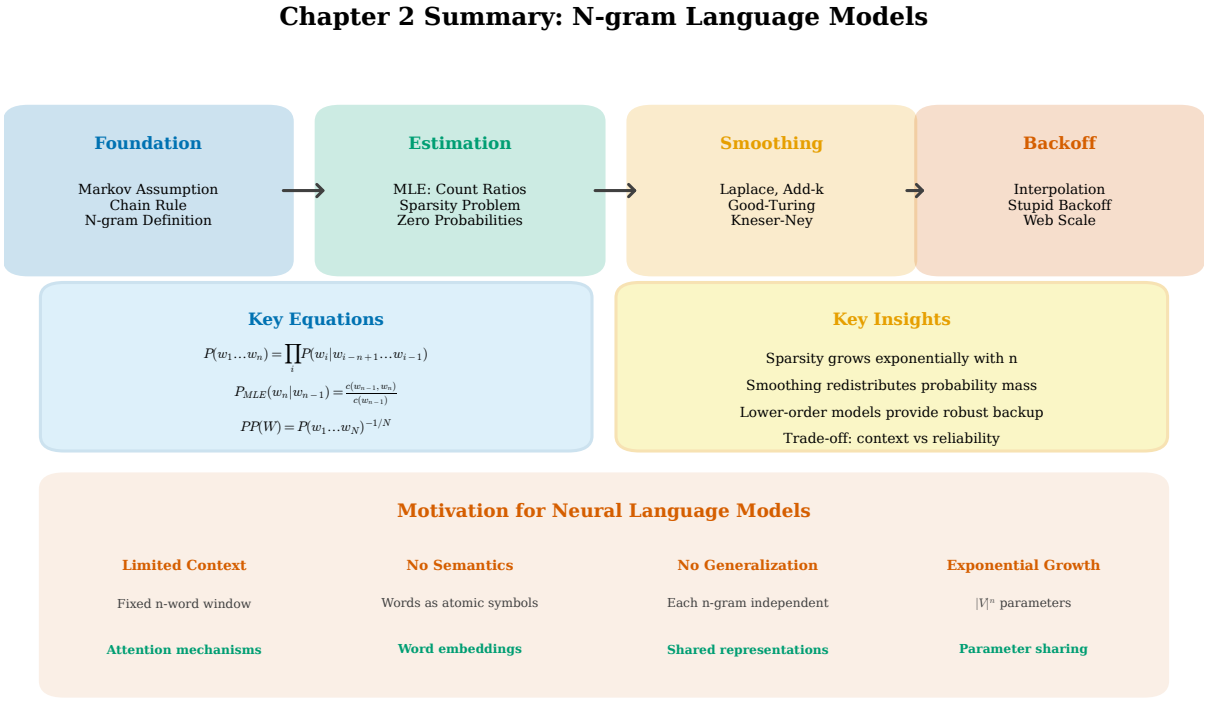

Figure 2.27: Visual summary of n-gram language model concepts. The chapter progresses from foundational concepts through estimation and smoothing to backoff strategies, culminating in an understanding of the limitations that motivate neural approaches.

2.8 Context Representation in N-gram Models

How This Chapter Represents Context
The fundamental question in language modeling is: How do we represent the context $w_1, \ldots, w_{t-1}$ to predict w_t?

- **Context representation**: N-gram models represent context as a fixed-length tuple of the preceding $n-1$ words
- **Context encoding**: The context is encoded implicitly through count table lookups—each unique context maps to a separate probability distribution
- **Limitation**: Contexts differing by a single word are treated as completely independent, with no sharing of statistical strength
- **Next chapter preview**: Neural language models will learn distributed context representations where similar contexts share parameters

N-gram models encode context in the simplest possible way: as an exact tuple of the preceding $n-1$ words, treated as an atomic lookup key with no internal structure. For a bigram model, the context is simply the previous word; for a trigram, the previous two words; and so forth, with each unique tuple serving as an independent index into a table of probability distributions. This representation has the virtue of extreme simplicity—we can look up probabilities in constant time using a hash table or trie indexed by the context tuple, and the resulting probabilities have clear interpretations as relative frequencies—but suffers from severe and fundamental limitations. Every unique context maps to a completely independent probability distribution, so the

model cannot recognize that "the black cat" and "the dark cat" share similar grammatical structure and should predict similar continuations. Changing even one word in the context creates an entirely new lookup key with no connection to related contexts. The exponential growth in the number of possible contexts ($|\mathcal{V}|^{n-1}$ for an n-gram model) means that the vast majority of possible contexts will never appear in any finite training corpus, forcing reliance on backoff and smoothing techniques rather than the direct estimation we would prefer. Neural language models, as we will see in subsequent chapters on word embeddings and recurrent networks, address these limitations by learning distributed representations of context as dense vectors in continuous space, where similar contexts occupy nearby points and therefore produce similar predictions, enabling the compositional generalization that n-gram models fundamentally cannot achieve.

We can now predict better because:

- The **Markov assumption** makes language modeling tractable by limiting context to $n-1$ words
- **Maximum likelihood estimation** from counts provides a simple, interpretable baseline
- **Smoothing techniques** redistribute probability mass to handle the **sparsity problem**
- **Kneser-Ney** captures word versatility through continuation counts, achieving state-of-the-art n-gram performance
- **Perplexity** provides a principled evaluation metric grounded in information theory

Next: Chapter **??** addresses how we define the vocabulary $\mathcal{V}$ itself through tokenization—the crucial preprocessing step that determines what units our language model predicts.

Exercises

1. **N-gram Counting and MLE.** Given the corpus: "`<s> the cat sat on the mat </s>`", "`<s> the dog sat on the floor </s>`", "`<s> the cat ran on the mat </s>`", list all unique bigrams with their counts, then compute $P_{\text{MLE}}(\text{mat}|\text{the})$ and $P_{\text{MLE}}(\text{sat}|\text{cat})$. Explain why $P(\text{bed}|\text{the}) = 0$ is problematic.

2. **Laplace Smoothing.** Using the corpus from Exercise 1 with vocabulary {`<s>`, `</s>`, the, cat, dog, sat, ran, on, mat, floor}, compute Laplace-smoothed probabilities $P_{\text{Laplace}}(\text{mat}|\text{the})$ and $P_{\text{Laplace}}(\text{bed}|\text{the})$. Discuss how add-one smoothing affects frequent versus rare bigrams.

3. **Perplexity Calculation.** A bigram model assigns: $P(\text{the}) = 0.10$, $P(\text{cat}|\text{the}) = 0.05$, $P(\text{sat}|\text{cat}) = 0.30$, $P(\text{</s>}|\text{sat}) = 0.20$. Calculate the sentence probability and perplexity. Express the result in bits per word.

4. **Sparsity Analysis.** For vocabulary size $|V| = 50,000$, compute the number of possible bigrams and trigrams. With a 1-billion-word corpus, estimate what fraction of possible trigrams will remain unobserved and explain why.

5. **Good-Turing Estimation.** Given frequency of frequencies $N_1 = 1000$, $N_2 = 500$, $N_3 = 300$, compute the Good-Turing adjusted count c^* for bigrams appearing once and twice. Calculate the probability mass allocated to unseen events.

6. **Kneser-Ney Insight.** "Francisco" appears 10,000 times (always after "San"); "Monday" appears 1,000 times (after many different words). For the novel context "next ___", explain which word should receive higher probability and why continuation counts matter.

7. **Backoff vs. Interpolation.** Describe how backoff and interpolation differ when estimating $P(\text{mat}|\text{the}, \text{floor})$ with zero trigram count. Provide a scenario where interpolation outperforms pure backoff.

8. **Cross-Entropy and Perplexity.** A model achieves 4 bits per word cross-entropy. Calculate the perplexity and bits per character (assuming 5 characters per word). Compare to Shannon's estimate of 1 bit per character for English.

9. **Context Length Trade-offs.** For vocabulary 30,000 and corpus 100 million words, estimate sparsity for bigram, trigram, and 4-gram models. Recommend an n-gram order and explain how your choice would change with 10 billion words.

10. **Limitations and Neural Solutions.** Explain how neural language models address: (1) bounded context windows, (2) lack of semantic similarity between words, (3) exponential parameter growth, and (4) inability to generalize across unseen contexts.

11. **Stupid Backoff.** With $\alpha = 0.4$, compute the score for 5-gram "the quick brown fox jumps" (count 1000, context count 5000). If unseen, compute the score using 4-gram "quick brown fox jumps" with relative frequency 0.001. Explain why scores differ from probabilities.

Bibliography

Frederick Jelinek. *Self-organized language modeling for speech recognition*. Morgan Kaufmann, 1990.

Index