This decomposition is justified by the chain rule of probability. The probability distribution over the initial sequence $P(x_1, \ldots, x_{n-1})$ may be modeled by a different model with a smaller value of n.

Training n-gram models is straightforward because the maximum likelihood estimate can be computed simply by counting how many times each possible n gram occurs in the training set. Models based on n-grams have been the core building block of statistical language modeling for many decades (Jelinek and Mercer, 1980; Katz, 1987; Chen and Goodman, 1999).

For small values of n, models have particular names: **unigram** for n=1, **bigram** for n=2, and **trigram** for n=3. These names derive from the Latin prefixes for the corresponding numbers and the Greek suffix "-gram" denoting something that is written.

Usually we train both an n-gram model and an $n-1$ gram model simultaneously. This makes it easy to compute

$$P(x_t \mid x_{t-n+1}, \ldots, x_{t-1}) = \frac{P_n(x_{t-n+1}, \ldots, x_t)}{P_{n-1}(x_{t-n+1}, \ldots, x_{t-1})} \tag{12.6}$$

simply by looking up two stored probabilities. For this to exactly reproduce inference in P_n, we must omit the final character from each sequence when we train P_{n-1}.

As an example, we demonstrate how a trigram model computes the probability of the sentence "`THE DOG RAN AWAY`." The first words of the sentence cannot be handled by the default formula based on conditional probability because there is no context at the beginning of the sentence. Instead, we must use the marginal probability over words at the start of the sentence. We thus evaluate $P_3(\texttt{THE DOG RAN})$. Finally, the last word may be predicted using the typical case, of using the conditional distribution $P(\texttt{AWAY} \mid \texttt{DOG RAN})$. Putting this together with equation 12.6, we obtain:

$$P(\texttt{THE DOG RAN AWAY}) = P_3(\texttt{THE DOG RAN})P_3(\texttt{DOG RAN AWAY})/P_2(\texttt{DOG RAN}). \tag{12.7}$$

A fundamental limitation of maximum likelihood for n-gram models is that P_n as estimated from training set counts is very likely to be zero in many cases, even though the tuple $(x_{t-n+1}, \ldots, x_t)$ may appear in the test set. This can cause two different kinds of catastrophic outcomes. When P_{n-1} is zero, the ratio is undefined, so the model does not even produce a sensible output. When P_{n-1} is non-zero but P_n is zero, the test log-likelihood is $-\infty$. To avoid such catastrophic outcomes, most n-gram models employ some form of **smoothing**. Smoothing techniques