This lecture is about **language models**, the technical definition of which is "models that assign probability to sentences."

You should be familiar with the concept of a probability distribution. The new twist is that we are assigning probability to infinitely many events (Σ^*). If you're not really comfortable with probability, this lecture might be helpful, since it gives you some concrete probability models to think about. If this lecture really floors you, please come to office hours or make an appointment.

A **random variable** is a variable that can take different values, due to chance. We usually use a capital letter to denote a random variable (e.g., A) and lower-case letters to denote values it can take (a). We use $p(A = a)$ to denote the probability that random variable A takes value a.

If random variables A and B are independent, then $p(A, B) = p(A) \times p(B)$. For any random variables $X_1, X_2, \ldots, X_\ell$, it is *always* true that:

$$p(X_1, X_2, \ldots, X_\ell) = p(X_1) \times p(X_2 \mid X_1) \times \cdots \times p(X_\ell \mid X_1, X_2, \ldots, X_{\ell-1})$$

This is the **chain rule** for probabilities. Note that it holds no matter how you order the events. Aside: we're abusing notation when we write "$p(X)$." Remember, we should really write $p(X = x)$, where X is the random variable and x is a value it can take.

Two extreme views of event sequences are that the events are *entirely* independent at each step (like coin flips) or that each event depends on the *entire* history.

Now, consider that every word in a sequence (sentence or document) is an event (i.e., the outcome of some random variable). There are relative merits to both the **unigram** model (every word independent of all the others) and the **full history** model (every word depends on the whole history). Basically: the unigram model generalizes too much, but the full history model only memorizes the data it was estimated from. However, note that expressions computing the probability of a sentence have the **same number of terms** in both models.

A middle ground between these two extremes are ***n*-gram models**. They predict each word based on the most recent history of $n - 1$ words (n is a parameter chosen by you; for words in natural language, n is often in $\{3, 4, 5\}$). This is called a **Markov assumption**. As n gets larger, the text the model generates looks better and better (or at least more like the training data). However, in the limit as $n \to \infty$, this becomes equivalent to the full history model.

Another key idea in this lecture: probability models need to be **learned from data**. For n-gram models, we need text corpora (plural of *corpus*). A simple way of "learning" or "training" the model is to **count** and **normalize** the counts into relative frequencies (out of all the times I see the word *San*, what proportion of the time is it followed by *Francisco*?).

A standard measurement of a language model's ability to predict unseen data (here, $\boldsymbol{w} = \langle w_1, w_2, \ldots, w_{|\boldsymbol{w}|} \rangle$) is **perplexity**:

$$\text{perplexity}(p(\cdot); \boldsymbol{w}) = 2^{-\frac{\log_2 p(\boldsymbol{w})}{|\boldsymbol{w}|}} = p(\boldsymbol{w})^{-\frac{1}{|\boldsymbol{w}|}} = \sqrt[|\boldsymbol{w}|]{\frac{1}{p(\boldsymbol{w})}}$$

A *lower* perplexity is better. This is only meaningful when comparing proper probability distributions (i.e., that sum to one) with the same vocabulary. Consider what happens to perplexity when

(a) some of the $p(\text{word} \mid \text{history})$ terms are 0; (b) the history is ignored and $p(\text{word})$ is uniform; or (c) we use full history and evaluate on the training data.

Four kinds of data: **training**, **held-out** (or **dev**), **devtest**, and **test**.

Maximum likelihood ("relative frequency") estimation:

$$\hat{p}_{\text{MLE}}(w_N \mid \langle w_1, \ldots, w_{N-1} \rangle) = \frac{\text{count}(\langle w_1, w_2, \ldots, w_{N-1}, w_N \rangle)}{\sum_{v \in \Sigma} \text{count}(\langle w_1, w_2, \ldots, w_{N-1}, v \rangle)}$$

Add-λ smoothing (commonly, $\lambda = 1$, in which case this is also called Laplace smoothing):

$$\hat{p}_{\text{add-}\lambda}(w_N \mid \langle w_1, \ldots, w_{N-1} \rangle) = \frac{\lambda + \text{count}(\langle w_1, w_2, \ldots, w_{N-1}, w_N \rangle)}{\lambda|\Sigma| + \sum_{v \in \Sigma} \text{count}(\langle w_1, w_2, \ldots, w_{N-1}, v \rangle)}$$

Good–Turing discounted counts:

$$c^* = \frac{(c+1) \times N_{c+1}}{N_c}$$

Linear interpolation ("mixture model"):

$$\begin{aligned}
\hat{p}_{\text{interp}}(w_N \mid \langle w_1, \ldots, w_{N-1} \rangle) = & \lambda_N \times \hat{p}_{\text{MLE}}(w_N \mid \langle w_1, \ldots, w_{N-1} \rangle) \\
& + \lambda_{N-1} \times \hat{p}_{\text{MLE}}(w_N \mid \langle w_2, \ldots, w_{N-1} \rangle) \\
& \vdots \\
& + \lambda_2 \times \hat{p}_{\text{MLE}}(w_N \mid \langle w_{N-1} \rangle) \\
& + \lambda_1 \times \hat{p}_{\text{MLE}}(w_N) \\
& + \lambda_0 \times \frac{1}{|\Sigma|}
\end{aligned}$$

Katz backoff (uses Good–Turing discounted counts!):

$$\hat{p}_{\text{Katz}}(w_N \mid \langle w_1, \ldots, w_{N-1} \rangle) = \begin{cases} \hat{p}_{\text{bo}}(w_N \mid \langle w_1, \ldots, w_{N-1} \rangle), & \text{if } \text{count}(\langle w_1, \ldots, w_N \rangle) > 0 \\ \hat{p}_{\text{Katz}}(w_N \mid \langle w_2, \ldots, w_{N-1} \rangle), & \text{if } \text{count}(\langle w_1, \ldots, w_{N-1} \rangle) = 0 \\ \alpha_{\langle w_1, \ldots, w_{N-1} \rangle} \times & \\ \quad \hat{p}_{\text{Katz}}(w_N \mid \langle w_2, \ldots, w_{N-1} \rangle), & \text{otherwise} \end{cases}$$

$$\hat{p}_{\text{bo}}(w_N \mid \langle w_1, \ldots, w_{N-1} \rangle) = \begin{cases} \frac{\text{count}^*(\langle w_1, \ldots, w_N \rangle)}{\text{count}(\langle w_1, \ldots, w_{N-1} \rangle)}, & \text{if } \text{count}(\langle w_1, \ldots, w_{N-1} \rangle) > 0 \\ 0, & \text{otherwise} \end{cases}$$

$$\alpha_{\langle w_1, \ldots, w_{N-1} \rangle} = \frac{1 - \sum_{v \in \Sigma : \text{count}(\langle w_1, \ldots, w_{N-1} v \rangle) > 0} \hat{p}_{\text{bo}}(v \mid \langle w_1, \ldots, w_{N-1} \rangle)}{1 - \sum_{v \in \Sigma : \text{count}(\langle w_1, \ldots, w_{N-1} v \rangle) > 0} \hat{p}_{\text{bo}}(v \mid \langle w_2, \ldots, w_{N-1} \rangle)}$$

Language modeling toolkits: SRILM, Cambridge-CMU, IRSTLM, KenLM. All are freely available and have similar functionality. *You may not use toolkits on the language modeling assignment, but you are welcome to use them on your project.*

See section 4.11 (pp. 117–119) for some details on how people have tried to estimate the entropy of English.