This lecture sets up two important ideas. The first is the use of **dynamic programming** to calculate the minimum number of single-character revisions to transform one string into another (the "minimum edit distance" problem). There are a number of variations on this problem; the simplest is the Levenshtein distance, which permits insertion, deletion, and substitution of characters to transform a source string $\boldsymbol{s}$ to a target string $\boldsymbol{t}$.

$$
\begin{aligned}
D_{0,0} &= 0 \\
D_{i,j} &= \min \begin{cases} D_{i-1,j} + \text{inscost}(t_i) \\ D_{i,j-1} + \text{delcost}(s_j) \\ D_{i-1,j-1} + \text{substcost}(t_i, s_j) \end{cases}
\end{aligned}
$$

The minimum edit (Levenshtein) distance is $D_{|\boldsymbol{t}|,|\boldsymbol{s}|}$. There are many generalizations, and this can be implemented using (weighted) finite-state transducers or a 2-dimensional array.

The second big concept in today's lecture is the **noisy channel model**. This is really more of a modeling paradigm than a specific model. It used to be called the "Bayesian" approach because it uses Bayes' rule, but that usually means something else in today's terminology. The noisy channel model assumes we want to recover the most likely system output y for an observed input x by modeling the probability distribution $p(y \mid x)$. To derive the noisy channel model, we use Bayes' rule, along with the fact that the marginal $p(x)$ is constant for a given x:

$$
\begin{aligned}
y^* &= \arg\max_y p(y \mid x) \\
&= \arg\max_y \frac{p(x \mid y) \times p(y)}{p(x)} && \text{Bayes' rule} \\
&= \arg\max_y \frac{p(x \mid y) \times p(y)}{\sum_{y'} p(x \mid y') \times p(y')} \\
&= \arg\max_y p(x \mid y) \times p(y)
\end{aligned}
$$

The resulting two factors are called, respectively, the **channel probability** (also called the *likelihood*) and the **source probability** (also called the *prior*). The model's view is that, first, a value of y is chosen according to the source model. Second, y passes through a channel that "garbles" it into x in a stochastic fashion; the channel model models that stochastic process. To choose y in our system, we decode the observed value of x. This usually means solving the arg max problem above. A good y is one that makes x likely but is also itself likely to occur in pure, ungarbled data.

We talked about using a noisy channel model for single-word spelling correction. This model can also be used in machine translation and speech recognition. In both cases, the source model is usually a language (e.g., N-gram) model. The channel model in Chinese→English machine translation is called a **translation model**; it turns English sentences (y) into Chinese sentences (x; seems backwards!). The channel model in speech recognition is called an **acoustic model**; it turns English word sequences (y) into acoustic signals (x; seems backwards!).