# Edit Distance, Spelling Correction, and the Noisy Channel

CS 585, Fall 2015
Introduction to Natural Language Processing
http://people.cs.umass.edu/~brenocon/inlp2015/

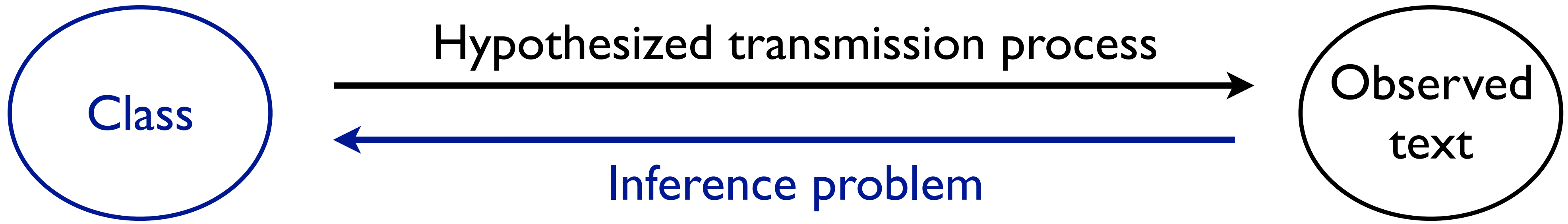
Brendan O'Connor

UMASS CS
SCHOOL OF COMPUTER SCIENCE
50 YEARS

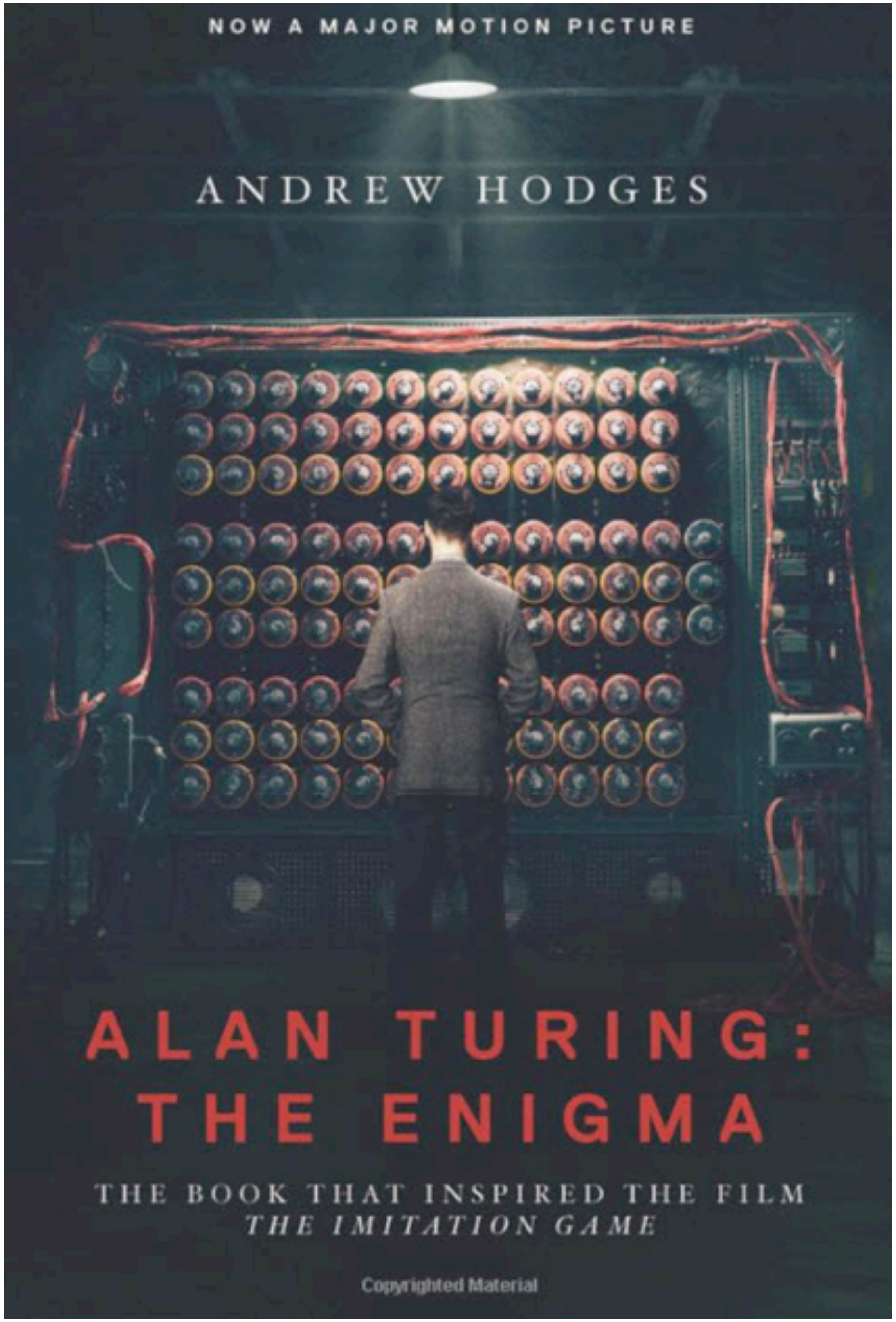
- Projects
- OH after class
- Some major topics in second half of course
  - Translation: spelling, machine translation
  - Syntactic parsing:  dependencies, hierarchical phrase structures
  - Coreference
  - Lexical semantics
  - Unsupervised language learning
  - Topic models, exploratory analysis?
  - Neural networks?

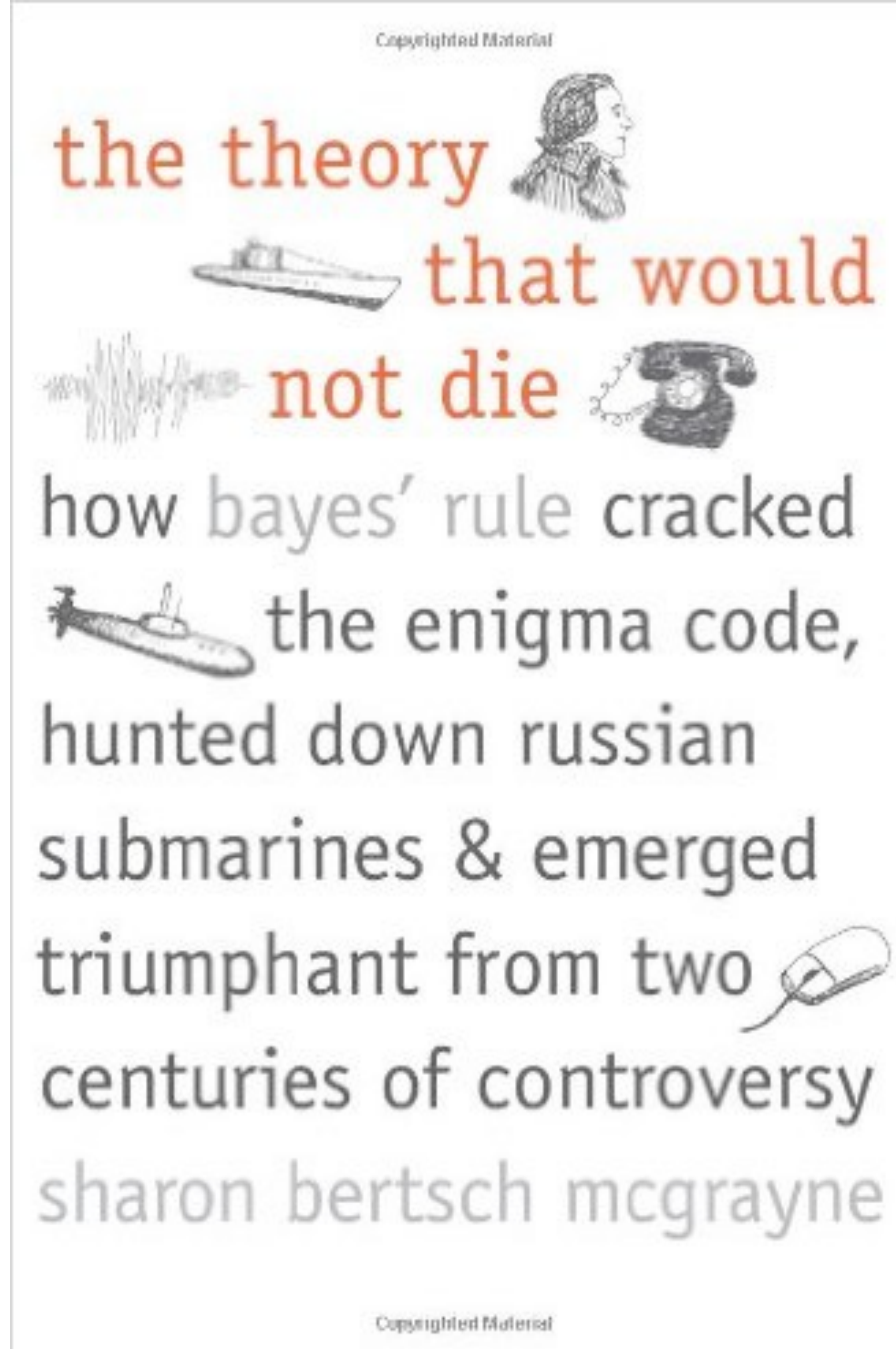
# Bayes Rule for doc classif.

Noisy
channel
model

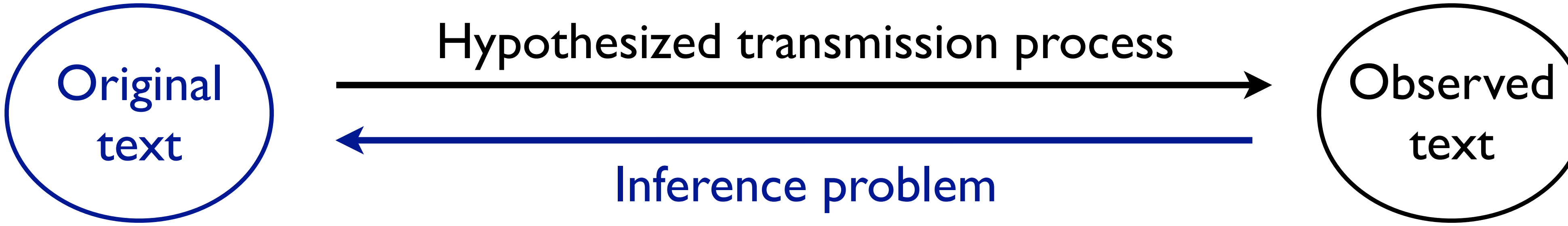



Previously:
Naive Bayes


NOW A MAJOR MOTION PICTURE
ANDREW HODGES
ALAN TURING: THE ENIGMA
THE BOOK THAT INSPIRED THE FILM
THE IMITATION GAME
Copyrighted Material


Copyrighted Material
the theory that would not die
how bayes' rule cracked the enigma code, hunted down russian submarines & emerged triumphant from two centuries of controversy
sharon bertsch mcgrayne
Copyrighted Material

# Noisy channel model




## Codebreaking

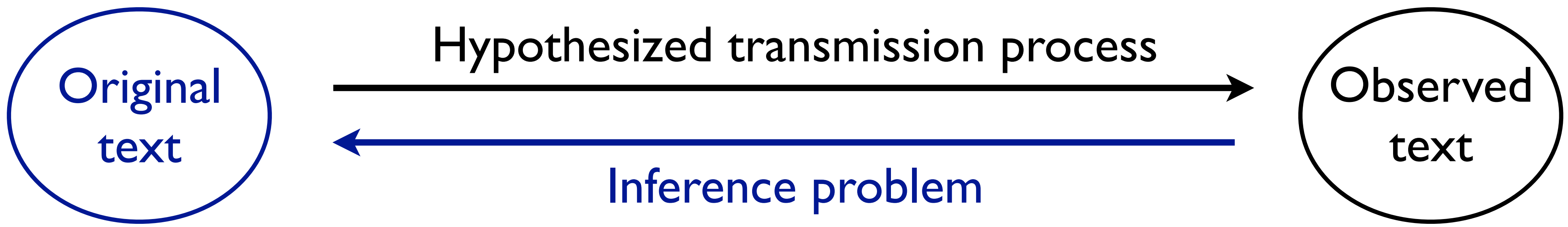
P(plaintext | encrypted text) $\propto$ P(encrypted text | plaintext) P(plaintext)

↓

INFERENCE:
Bletchley Park (WWII)


TRANSMISSION:
Enigma machine

# Noisy channel model

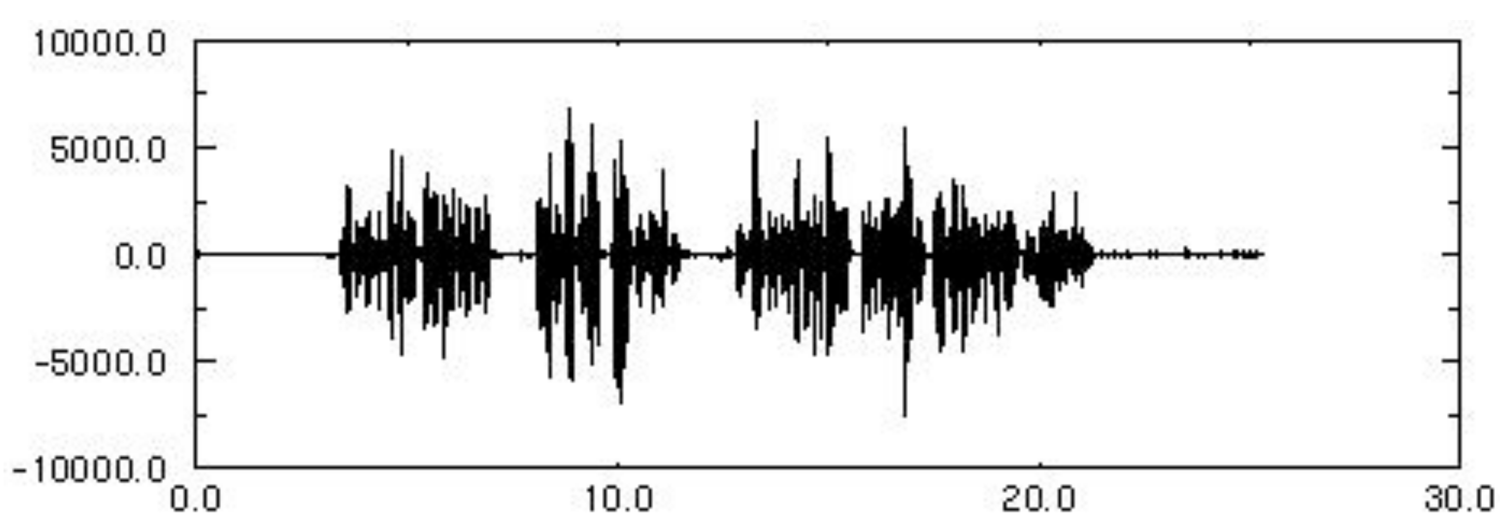



## Codebreaking

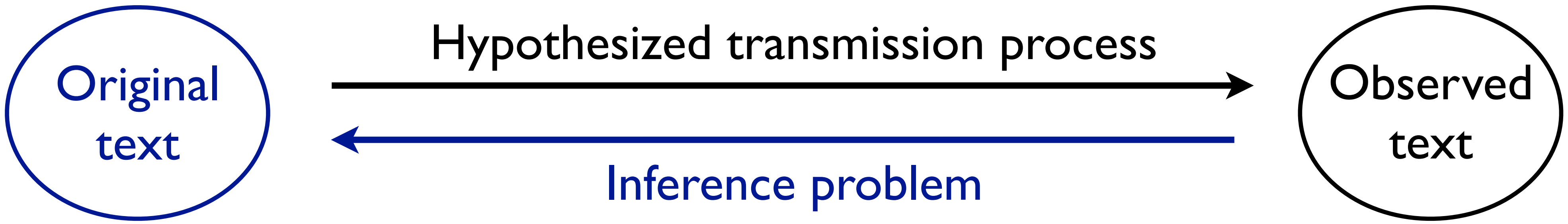
P(plaintext | encrypted text) $\propto$ P(encrypted text | plaintext) P(plaintext)

## Speech recognition

P(text | acoustic signal) $\propto$ P(acoustic signal | text) P(text)



# Noisy channel model




## Codebreaking

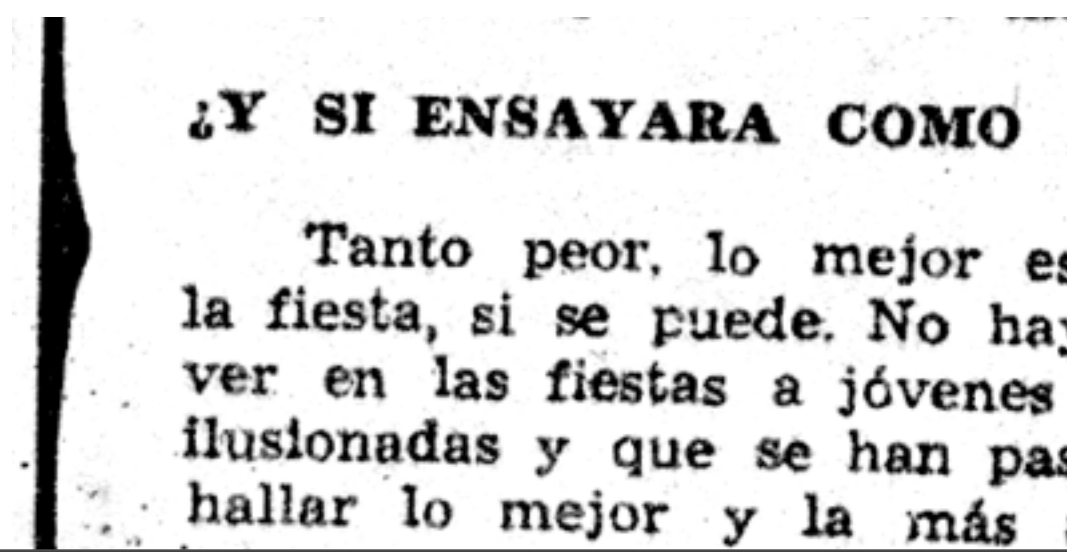
$$P(\text{plaintext} \mid \text{encrypted text}) \propto P(\text{encrypted text} \mid \text{plaintext})\, P(\text{plaintext})$$

## Speech recognition

$$P(\text{text} \mid \text{acoustic signal}) \propto P(\text{acoustic signal} \mid \text{text})\, P(\text{text})$$

## Optical character recognition

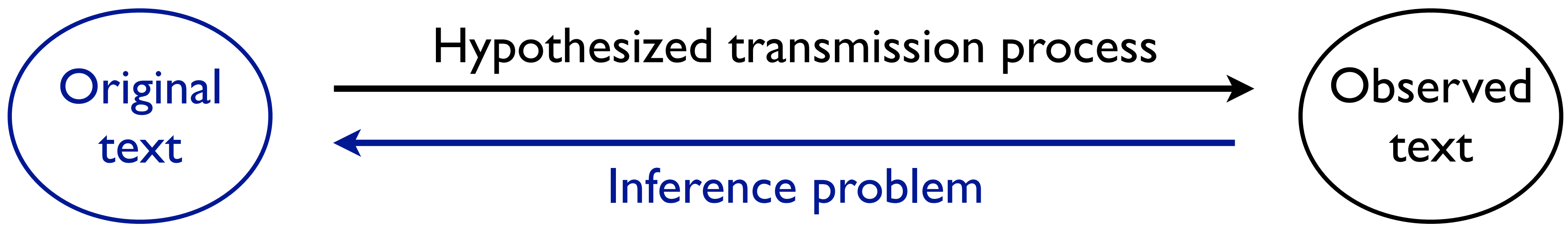
$$P(\text{text} \mid \text{image}) \propto P(\text{image} \mid \text{text})\, P(\text{text})$$



¿Y SI ENSAYARA COMO

Tanto peor, lo mejor es
la fiesta, si se puede. No hay
ver en las fiestas a jóvenes
ilusionadas y que se han pas
hallar lo mejor y la más

# Noisy channel model




## Codebreaking

$P(\text{plaintext} \mid \text{encrypted text}) \propto P(\text{encrypted text} \mid \text{plaintext})\, P(\text{plaintext})$

## Speech recognition

$P(\text{text} \mid \text{acoustic signal}) \propto P(\text{acoustic signal} \mid \text{text})\, P(\text{text})$

## Optical character recognition

$P(\text{text} \mid \text{image}) \propto P(\text{image} \mid \text{text})\, P(\text{text})$

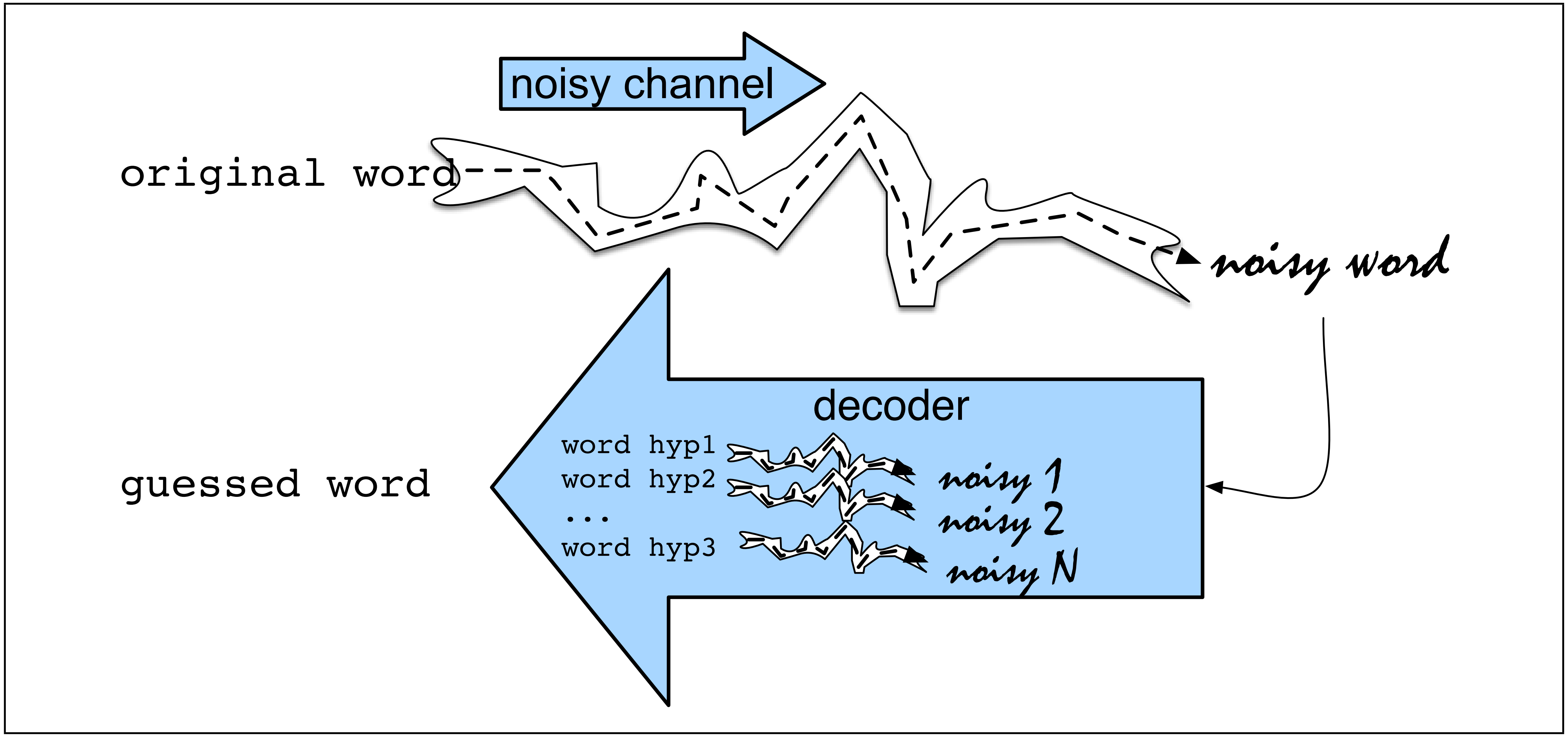
## Machine translation

$P(\text{target text} \mid \text{source text}) \propto P(\text{source text} \mid \text{target text})\, P(\text{target text})$

## Spelling correction

$P(\text{target text} \mid \text{source text}) \propto P(\text{source text} \mid \text{target text})\, P(\text{target text})$

# Noisy channel model



# Spelling correction as noisy channel

I was too tired to go
I was to tired to go
I was zzz tired to go
...

Hypothetical Model →

← Inference problem

INPUT
I was ti tired to go

$$\hat{w} = \underset{w \in C}{\operatorname{argmax}} \overbrace{P(x|w)}^{\text{channel model}} \overbrace{P(w)}^{\text{prior}}$$

Edit distance

Language model

1. How to score possible translations?
2. How to efficiently search over them?

# Edit distance

- Tasks
  - Calculate numerical similarity between pairs
    - President Barack Obama
    - President Barak Obama
  - Enumerate edits with distance=1

- Model: Assume possible changes.
  - Deletions: actress => acress
  - Insertions: cress => acress
  - Substitutions: access => acress
  - [Transpositions: caress => acress]

- Probabilistic model: assume each has prob of occurrence

# Levenshtein Algorithm

- Goal: Infer minimum edit distance, and argmin edit path, for a pair of strings. e.g.: (*intention*, *execution*)

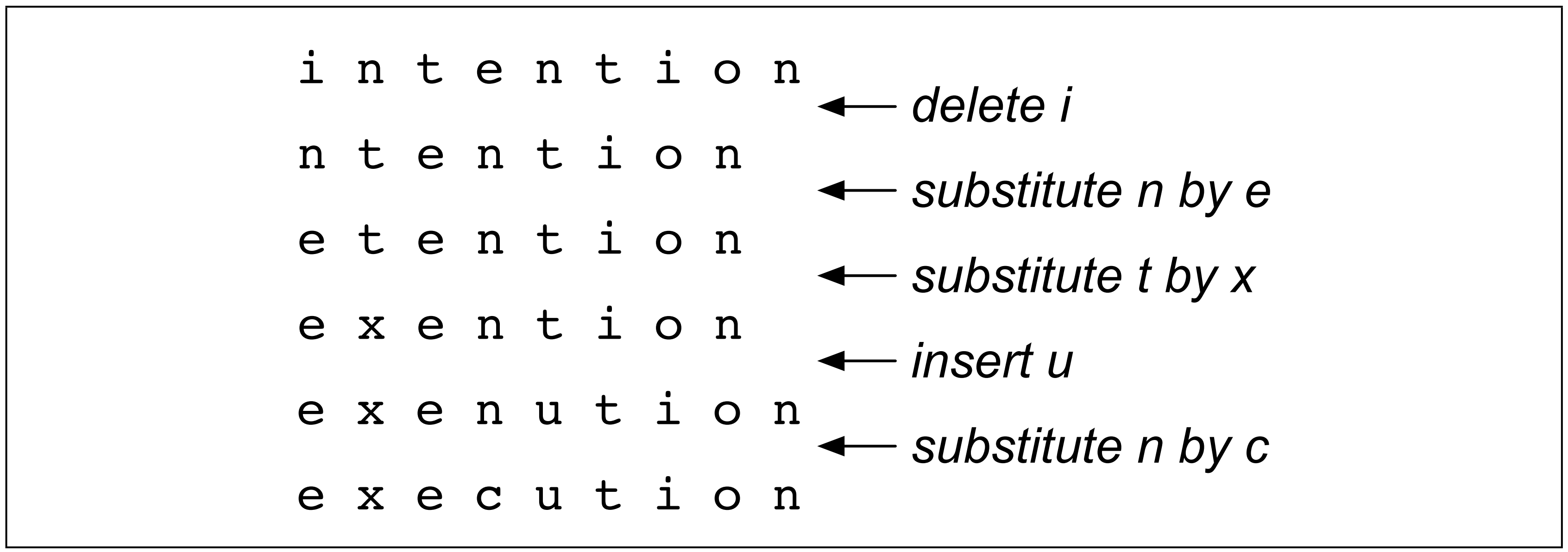



**Figure 2.14** Path from *intention* to *execution*.

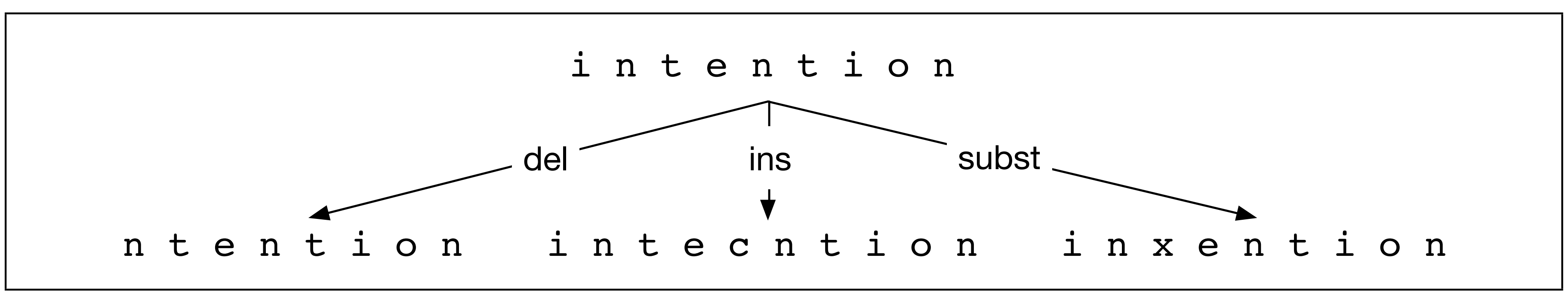



**Figure 2.13** Finding the edit distance viewed as a search problem

# [added after lecture]

- Want to calculate:
  Minimum edit distance between two strings X,Y, lengths n,m

Declaratively, edit distance has a recursive substructure:
d(barak obama, barack obama) = d(barak obama, barack obam) + InsertionCost

This allows for a **dynamic programming** algorithm to quickly compute the lowest cost path -- specifically, the **Levenshtein algorithm**.
(We'll just do the version with ins/del/subst, no transpositions)

# Levenshtein Algorithm

- Want to calculate:
  Minimum edit distance between two strings X,Y, lengths n,m
- D(i,j): edit dist between X[1..i] and Y[1..j].
  D(n,m): edit dist between X and Y

$$D[i,j] = \min \begin{cases} D[i-1,j] + \text{del-cost}(source[i]) \\ D[i,j-1] + \text{ins-cost}(target[j])) \\ D[i-1,j-1] + \text{sub-cost}(source[i], target[j]) \end{cases}$$

ins,del=1
sub=2

$$D[i,j] = \min \begin{cases} D[i-1,j] + 1 \\ D[i,j-1] + 1 \\ D[i-1,j-1] + \begin{cases} 2; & \text{if } source[i] \neq target[j] \\ 0; & \text{if } source[i] = target[j] \end{cases} \end{cases}$$

- Levenshtein algorithm: dynamic programming algorithm
  to quickly calculate all D[i,j].

| Src\Tar | # | e | x | e | c | u | t | i | o | n |
|---|---|---|---|---|---|---|---|---|---|---|
| # | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| i | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 6 | 7 | 8 |
| n | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 7 | 8 | 7 |
| t | 3 | 4 | 5 | 6 | 7 | 8 | 7 | 8 | 9 | 8 |
| e | 4 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 9 |
| n | 5 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 10 |
| t | 6 | 5 | 6 | 7 | 8 | 9 | 8 | 9 | 10 | 11 |
| i | 7 | 6 | 7 | 8 | 9 | 10 | 9 | 8 | 9 | 10 |
| o | 8 | 7 | 8 | 9 | 10 | 11 | 10 | 9 | 8 | 9 |
| n | 9 | 8 | 9 | 10 | 11 | 12 | 11 | 10 | 9 | 8 |

**Figure 2.16** Computation of minimum edit distance between *intention* and *execution* with the algorithm of Fig. 2.15, using Levenshtein distance with cost of 1 for insertions or deletions, 2 for substitutions. In italics are the initial values representing the distance from the empty string.

# Backpointers

| | # | e | x | e | c | u | t | i | o | n |
|---|---|---|---|---|---|---|---|---|---|---|
| **#** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| **i** | **1** | ↖←↑ 2 | ↖←↑ 3 | ↖←↑ 4 | ↖←↑ 5 | ↖←↑ 6 | ↖←↑ 7 | ↖ 6 | ← 7 | ← 8 |
| **n** | 2 | ↖←↑ **3** | ↖←↑ 4 | ↖←↑ 5 | ↖←↑ 6 | ↖←↑ 7 | ↖←↑ 8 | ↑ 7 | ↖←↑ 8 | ↖ 7 |
| **t** | 3 | ↖←↑ 4 | ↖←↑ **5** | ↖←↑ 6 | ↖←↑ 7 | ↖←↑ 8 | ↖ 7 | ←↑ 8 | ↖←↑ 9 | ↑ 8 |
| **e** | 4 | ↖ 3 | ← 4 | ↖← **5** | ← **6** | ← 7 | ←↑ 8 | ↖←↑ 9 | ↖←↑ 10 | ↑ 9 |
| **n** | 5 | ↑ 4 | ↖←↑ 5 | ↖←↑ 6 | ↖←↑ 7 | ↖←↑ **8** | ↖←↑ 9 | ↖←↑ 10 | ↖←↑ 11 | ↖↑ 10 |
| **t** | 6 | ↑ 5 | ↖←↑ 6 | ↖←↑ 7 | ↖←↑ 8 | ↖←↑ 9 | ↖ **8** | ← 9 | ← 10 | ←↑ 11 |
| **i** | 7 | ↑ 6 | ↖←↑ 7 | ↖←↑ 8 | ↖←↑ 9 | ↖←↑ 10 | ↑ 9 | ↖ **8** | ← 9 | ← 10 |
| **o** | 8 | ↑ 7 | ↖←↑ 8 | ↖←↑ 9 | ↖←↑ 10 | ↖←↑ 11 | ↑ 10 | ↑ 9 | ↖ **8** | ← 9 |
| **n** | 9 | ↑ 8 | ↖←↑ 9 | ↖←↑ 10 | ↖←↑ 11 | ↖←↑ 12 | ↑ 11 | ↑ 10 | ↑ 9 | ↖ **8** |

**Figure 2.17** When entering a value in each cell, we mark which of the three neighboring cells we came from with up to three arrows. After the table is full we compute an **alignment** (minimum edit path) by using a **backtrace**, starting at the **8** in the lower-right corner and following the arrows back. The sequence of bold cells represents one possible minimum cost alignment between the two strings.

# Dynamic programming

In his autobiography Bellman (1984) explains how he originally came up with the term *dynamic programming*:

> "...The 1950s were not good years for mathematical research. [the] Secretary of Defense ...had a pathological fear and hatred of the word, research... I decided therefore to use the word, "programming". I wanted to get across the idea that this was dynamic, this was multi-stage... I thought, let's ... take a word that has an absolutely precise meaning, namely dynamic... it's impossible to use the word, dynamic, in a pejorative sense. Try thinking of some combination that will possibly give it a pejorative meaning. It's impossible. Thus, I thought dynamic programming was a good name. It was something not even a Congressman could object to."

# Dynamic programming

In his autobiography Bellman (1984) explains how he originally came up with the term *dynamic programming*:

> "...The 1950s were not good years for mathematical research. [the] Secretary of Defense ...had a pathological fear and hatred of the word, research... I decided therefore to use the word, "programming". I wanted to get across the idea that this was dynamic, this was multi-stage... I thought, let's ... take a word that has an absolutely precise meaning, namely dynamic... it's impossible to use the word, dynamic, in a pejorative sense. Try thinking of some combination that will possibly give it a pejorative meaning. It's impossible. Thus, I thought dynamic programming was a good name. It was something not even a Congressman could object to."

- Levenshtein, Viterbi are both examples

# Channel model

- Norvig reading: Just make up edit parameters: deletions, edits, etc. all have logprob = -1
- To estimate from data: Need access to a corpus of mistakes

**additional**: addional, additonal
**environments**: enviornments, enviorments, enviroments
**preceded**: preceeded
...

$$P(x|w) = \begin{cases} \frac{\text{del}[x_{i-1}, w_i]}{\text{count}[x_{i-1} w_i]}, \text{ if deletion} \\ \frac{\text{ins}[x_{i-1}, w_i]}{\text{count}[w_{i-1}]}, \text{ if insertion} \\ \frac{\text{sub}[x_i, w_i]}{\text{count}[w_i]}, \text{ if substitution} \\ \frac{\text{trans}[w_i, w_{i+1}]}{\text{count}[w_i w_{i+1}]}, \text{ if transposition} \end{cases}$$

$$\hat{w} = \underset{w \in C}{\operatorname{argmax}} \overbrace{P(x|w)}^{\text{channel model}} \overbrace{P(w)}^{\text{prior}}$$

Edit distance

Language model

| Candidate Correction | Correct Letter | Error Letter | x\|w | P(x\|w) |
|---|---|---|---|---|
| actress | t | - | c\|ct | .000117 |
| cress | - | a | a\|# | .00000144 |
| caress | ca | ac | ac\|ca | .00000164 |
| access | c | r | r\|c | .000000209 |
| across | o | e | e\|o | .0000093 |
| acres | - | s | es\|e | .0000321 |
| acres | - | s | ss\|s | .0000342 |

**Figure 6.4** Channel model for acress; the probabilities are taken from the *del*[], *ins*[], *sub*[], and *trans*[] confusion matrices as shown in Kernighan et al. (1990).

$$\hat{w} = \underset{w \in C}{\operatorname{argmax}} \overbrace{P(x|w)}^{\text{channel model}} \overbrace{P(w)}^{\text{prior}}$$

Edit distance — Language model

| Candidate Correction | Correct Letter | Error Letter | x\|w | P(x\|w) |
|---|---|---|---|---|
| actress | t | - | c\|ct | .000117 |
| cress | - | a | a\|# | .00000144 |
| caress | ca | ac | ac\|ca | .00000164 |
| access | c | r | r\|c | .000000209 |
| across | o | e | e\|o | .0000093 |
| acres | - | s | es\|e | .0000321 |
| acres | - | s | ss\|s | .0000342 |

**Figure 6.4** Channel model for `acress`; the probabilities are taken from the *del*[], *ins*[], *sub*[], and *trans*[] confusion matrices as shown in Kernighan et al. (1990).

## Unigram LM:

| w | count(w) | p(w) |
|---|---|---|
| actress | 9,321 | .0000231 |
| cress | 220 | .000000544 |
| caress | 686 | .00000170 |
| access | 37,038 | .0000916 |
| across | 120,844 | .000299 |
| acres | 12,874 | .0000318 |

$$\hat{w} = \underset{w \in C}{\operatorname{argmax}} \overbrace{P(x|w)}^{\text{channel model}} \overbrace{P(w)}^{\text{prior}}$$

Edit distance

Language model

| Candidate Correction | Correct Letter | Error Letter | x\|w | P(x\|w) |
|---|---|---|---|---|
| actress | t | - | c\|ct | .000117 |
| cress | - | a | a\|# | .00000144 |
| caress | ca | ac | ac\|ca | .00000164 |
| access | c | r | r\|c | .000000209 |
| across | o | e | e\|o | .0000093 |
| acres | - | s | es\|e | .0000321 |
| acres | - | s | ss\|s | .0000342 |

**Figure 6.4** Channel model for acress; the probabilities are taken from the *del*[], *ins*[], *sub*[], and *trans*[] confusion matrices as shown in Kernighan et al. (1990).

## Unigram LM:

| w | count(w) | p(w) |
|---|---|---|
| actress | 9,321 | .0000231 |
| cress | 220 | .000000544 |
| caress | 686 | .00000170 |
| access | 37,038 | .0000916 |
| across | 120,844 | .000299 |
| acres | 12,874 | .0000318 |

=> unnorm. posterior:

| Candidate Correction | Correct Letter | Error Letter | x\|w | P(x\|w) | P(w) | $10^9$*P(x\|w)P(w) |
|---|---|---|---|---|---|---|
| actress | t | - | c\|ct | .000117 | .0000231 | 2.7 |
| cress | - | a | a\|# | .00000144 | .000000544 | 0.00078 |
| caress | ca | ac | ac\|ca | .00000164 | .00000170 | 0.0028 |
| access | c | r | r\|c | .000000209 | .0000916 | 0.019 |
| across | o | e | e\|o | .0000093 | .000299 | 2.8 |
| acres | - | s | es\|e | .0000321 | .0000318 | 1.0 |
| acres | - | s | ss\|s | .0000342 | .0000318 | 1.0 |

**Figure 6.5** Computation of the ranking for each candidate correction, using the language model shown earlier and the error model from Fig. 6.4. The final score is multiplied by $10^9$ for readability.

# N-Gram Language Models

- *was called a "stellar and versatile acress whose combination of sass and glamour has defined her*
- More context helps!  Assume a bigram Markov model

P(acress | versatile _ whose) = ?

$$\begin{aligned} \text{P(actress|versatile)} &= .000021 \\ \text{P(across|versatile)} &= .000021 \\ \text{P(whose|actress)} &= .0010 \\ \text{P(whose|across)} &= .000006 \end{aligned}$$

$$\begin{aligned} \text{P(“versatile actress whose”)} &= .000021 * .0010 = 210 \times 10^{-10} \\ \text{P(“versatile across whose”)} &= .000021 * .000006 = 1 \times 10^{-10} \end{aligned}$$

- Many misspellings are legitimate English words.  Language model is key.

  Even assuming only 1 error per sentence:

$X = \texttt{Only two of thew apples}$

```
only two of thew apples
oily two of thew apples
only too of thew apples
only to of thew apples
only tao of the apples
only two on thew apples
only two off thew apples
only two of the apples
only two of threw apples
only two of thew applies
only two of thew dapples
...
```