# Statistical Language Models

- Introduction to Language Models
- Noisy Channel model
- Simple Markov Models
- Smoothing

# Introduction

- Statistical models of words of sentences – language models
- Probability of all possible sequences of words
- Inspired in speech recognition techniques
- Probability of next word based on previous

# Probability Theory (I)

- X be uncertain outcome of some event.
  - Represented as a random variable
- V(X)  finite number of possible outcome (not a real number)
- P(X=x), probability of the particular outcome x (x belongs V(X))
  - X desease of your patient, V(X) all possible diseases

# Probability Theory(II)

Conditional probability of the outcome of an event based upon the outcome of a second event

We pick two words randomly from a book. We know first word is **the**, we want to know probability second word is **dog**

$$P(W_2 = \textbf{dog} | W_1 = \textbf{the}) = | W_1 = \textbf{the}, W_2 = \textbf{dog} | / | W_1 = \textbf{the} |$$

Bayes's law: $P(x|y) = P(x)\, P(y|x) \,/\, P(y)$

# Probability Theory (III)

**Bayes's law: P(x|y) = P(x) P(y|x) / P(y)**

P(desease/symptom)=
P(desease)P(symptom/desease)/P(symptom)

P($w_{1,n}$| speech signal) = P($w_{1,n}$)P(speech signal | $w_{1,n}$)/
P(speech signal)

We only need to maximize the numerator

P(speech signal | $w_{1,n}$) expresses how well the speech signal fits the sequence of words $w_{1,n}$

# Statistical Model of a Language

Vocabulary (V), word $w \in V$

Language (L), sentence $s \in L$

$L \subset V^*$ usually infinite

$s = w_1, \ldots w_N$

- Probability of s $P(s)$
- For sequences of words of length n assign a number to $P(W_{1,n} = w_{1,n})$, being $w_{1,n}$ a sequence of words

# Ngram Model

- Simple but durable statistical model
- Useful to indentify words in noisy, ambigous input.
- Speech recognition, many input speech sounds similar and confusable
- Machine translation, spelling correction, handwriting recognition, predictive text input
- **Other NLP tasks: part of speech tagging, NL generation, word similarity**

# CORPORA

- Corpora (singular corpus) are online collections of text or speech.

- **Brown** Corpus: 1 million word collection from 500 written texts  from different genres (newspaper,novels, academic).

  - Punctuation can be treated as words.

- **Switchboard** corpus: 2430 Telephone conversations averaging 6 minutes each, 240 hour of speech and 3 million words"

# Training and Test Sets

- Probabilities of N-gram model come from the corpus it is trained for
- Data in the corpus is divided into training set (or training corpus) and test set (or test corpus).
- Perplexity: compare statistical models

# Ngram Model

- How can we compute probabilities of entire sequences $P(w_1,w_2,..,w_n)$

Descomposition using the chain rule of probability $P(w_1,w_2,..,w_n)$
$=P(w_1)P(w_2|w_1)P(w_3|w_1,w_2),\ldots P(w_n|w_1..,w_{n-1})$

- Assigns a conditional probability to possible next words considering the history.

- Markov assumption : we can predict the probability of some future unit without looking too far into the past.

- Bigrams only consider previous usint, trigrams, two previous unit, n-grams, n previous unit

# Ngram Model

- Assigns a conditional probability to possible next words.Only n-1 previous words have effect on the probabilities of next word

- For n = 3, Trigrams $P(w_n|w_1..,w_{n-1}) = P(w_n|w_{n-2},w_{n-1})$

- How we estimate these trigram or N-gram probabilities?

**To maximize the likelihood of the training set T given the model M --- P(T/M)**

- To create the model use training text (corpus), taking counts and normalizing them so they lie between 0 and 1.

# Ngram Model

- For n = 3, Trigrams

$$P(w_n|w_1..,w_{n-1}) = P(w_n|w_{n-2},w_{n-1})$$

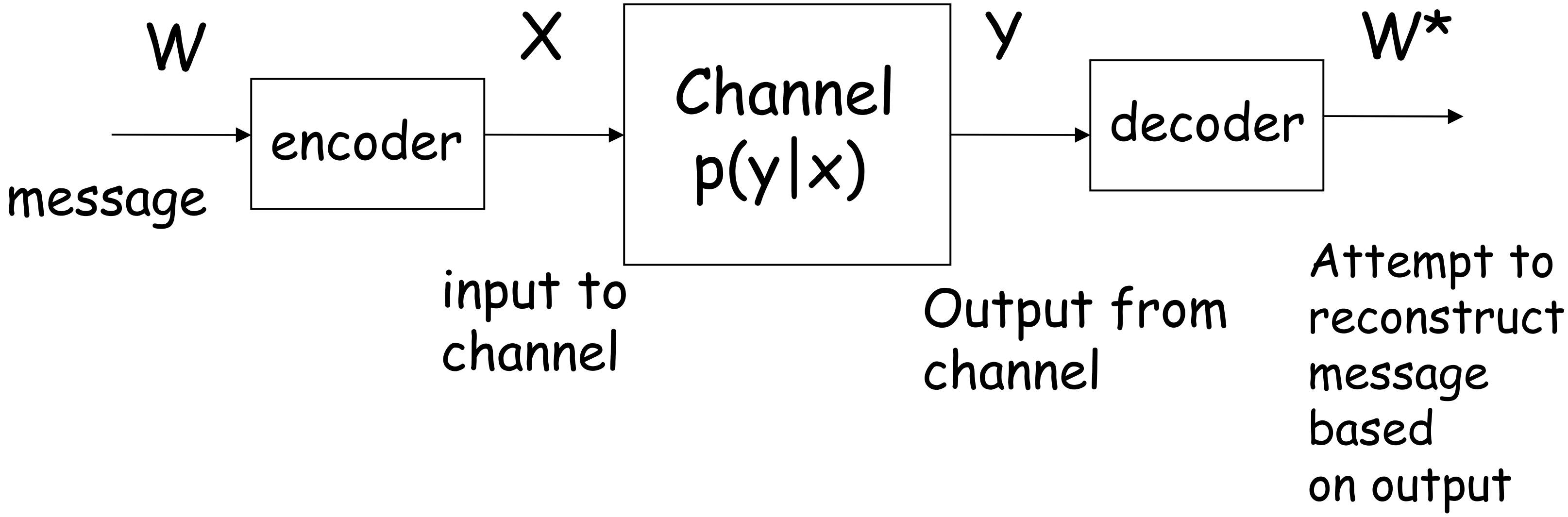
- To create the model use training text and record pairs and triples of words that appear in the text and how many times

$$P(w_i|w_{i-2},w_{i-1}) = C(w_{i-2,i}) / C(w_{i-2,i-1})$$

**P(submarine|the, yellow) = C(the,yellow, submarine)/C(the,yellow)**

**Relative frequency: observed frequency of a particular sequence divided by observed fequency of a prefix**

# Noisy Channel Model




In language processing the problem is reduced to decode for getting the most likely input given the output

Real Language X

Noisy channel X->Y

Observed Language Y

Correct text

Errors

Text with errors

**We want to retrieve X from Y**

**Real Language X**

**Correct text**

**Noisy channel X->Y**

**Space removing**

**Observed Language Y**

**Text without spaces**

**We want to retrieve X from Y**

**Real Language X**

**Noisy channel X->Y**

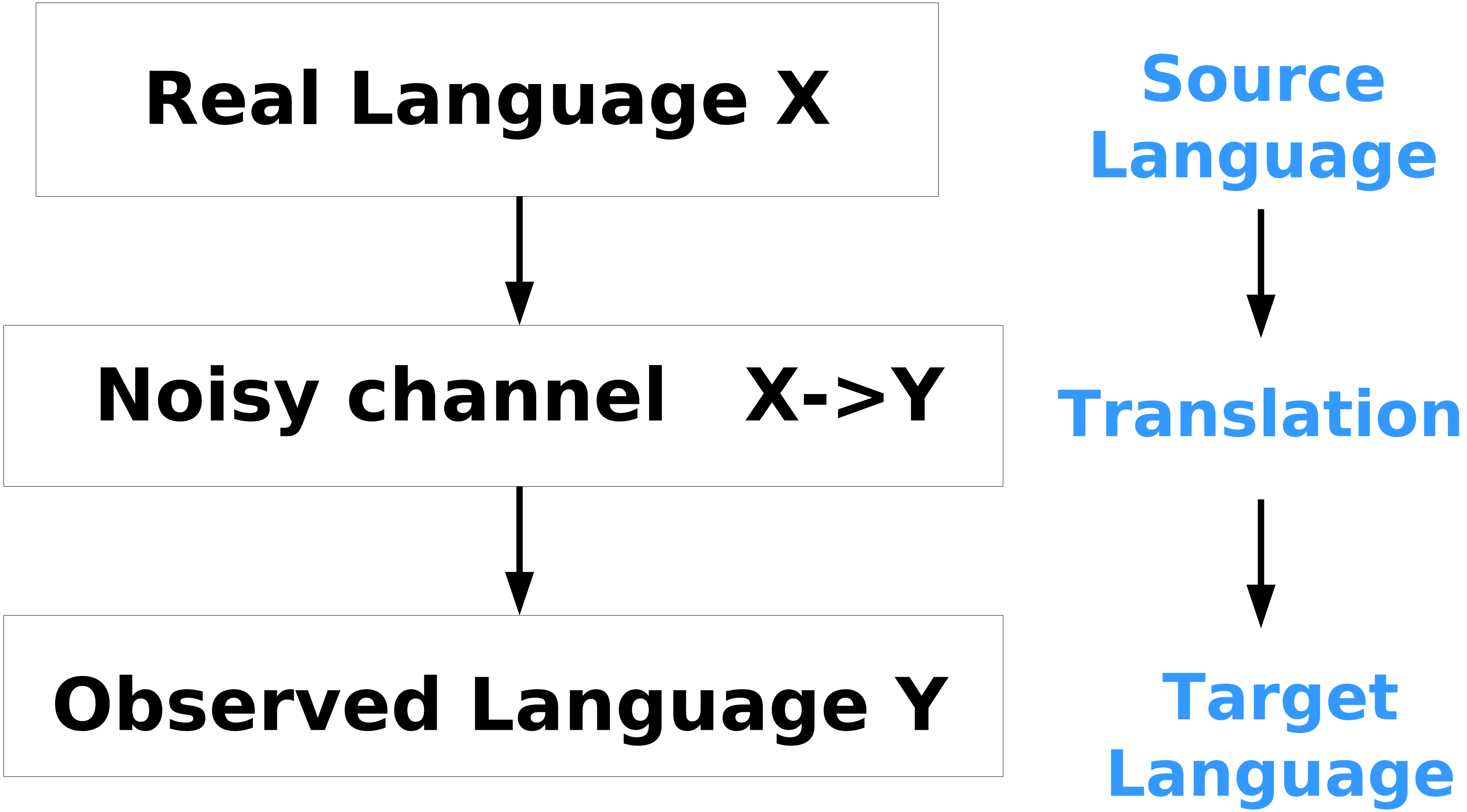
**Observed Language Y**

**Text**

**Text to speech generator**

**Speech**

**We want to retrieve X from Y**




**We want to retrieve X from Y**

# Example: ASR Automatic Speech Recognizer

Acoustic chain → word chain

$$X_1 \ldots X_T \longrightarrow \boxed{\text{ASR}} \longrightarrow w_1 \ldots w_N$$

Language model Acoustic Model

$$s_{OPT} = \underset{s}{\operatorname{argmax}}\, P(s \mid a) = \underset{s}{\operatorname{argmax}}\, P(s) \cdot P(a \mid s) = \underset{s}{\operatorname{argmax}}\, P(w_1^N) \cdot P(x_1^T \mid w_1^N)$$

# Example: Machine Translation

Target Language Model

Translation Model

$$o_{OPT} = \operatorname*{argmax}_{o} P(o \mid f) = \operatorname*{argmax}_{o} P(o) \cdot P(f \mid o)$$

- Naive Implementation
  - Enumerate s $\in$ L
  - Compute p(s)
  - Parameters of the model |L|
- But …
  - L is usually infinite
  - How to estimate the parameters?
- Simplifications
- History
  - $h_i = \{ w_i, \dots w_{i-1}\}$
- Markov Models

$$P(s) = P(w_1^N) = \prod_{i=1}^{N} P(w_i \mid h_i)$$

- Markov Models of order n + 1
  - $P(w_i|h_i) = P(w_i|w_{i-n+1}, \ldots w_{i-1})$
- 0-gram

$$\forall i \quad P(w_i) = \frac{1}{|V|}$$

- 1-gram
  - $P(w_i|h_i) = P(w_i)$
- 2-gram
  - $P(w_i|h_i) = P(w_i|w_{i-1})$
- 3-gram
  - $P(w_i|h_i) = P(w_i|w_{i-2},w_{i-1})$

- n large:
  - more context information (more discriminative power)
- n small:
  - more cases in the training corpus (more reliable)
- Selecting n:
  - ej. for |V| = 20.000

| ***n*** | **num. parameters** |
|---|---|
| **2 (bigrams)** | **400,000,000** |
| **3 (trigrams)** | **8,000,000,000,000** |
| **4 (4-grams)** | **1.6 x $10^{17}$** |

- Parameters of an n-gram model
  - $|V|^n$
- MLE estimation
  - From a training corpus
- Problem of sparseness

- 1-gram Model

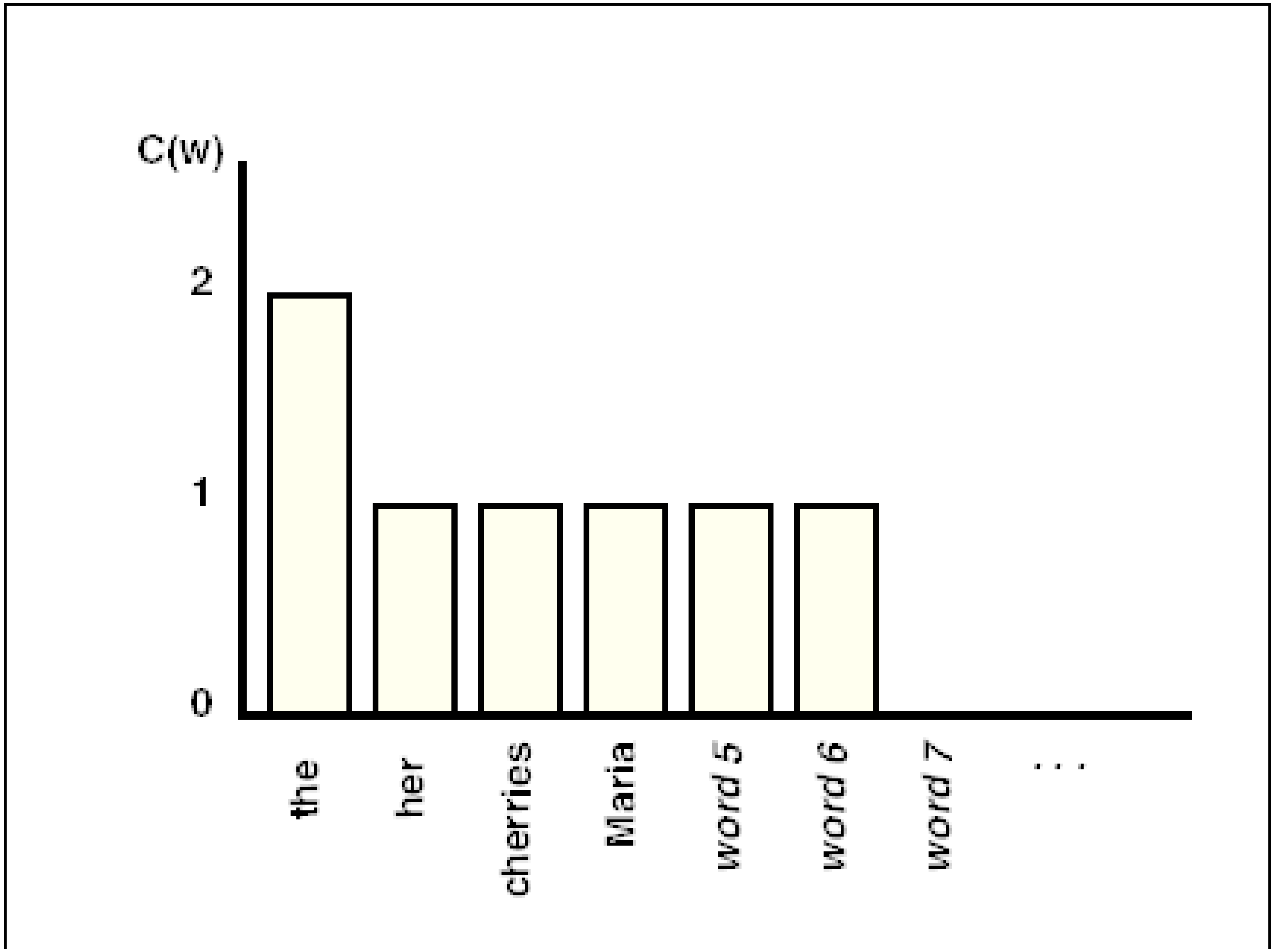
$$P_{MLE}(w) = \frac{C(w)}{|V|}$$

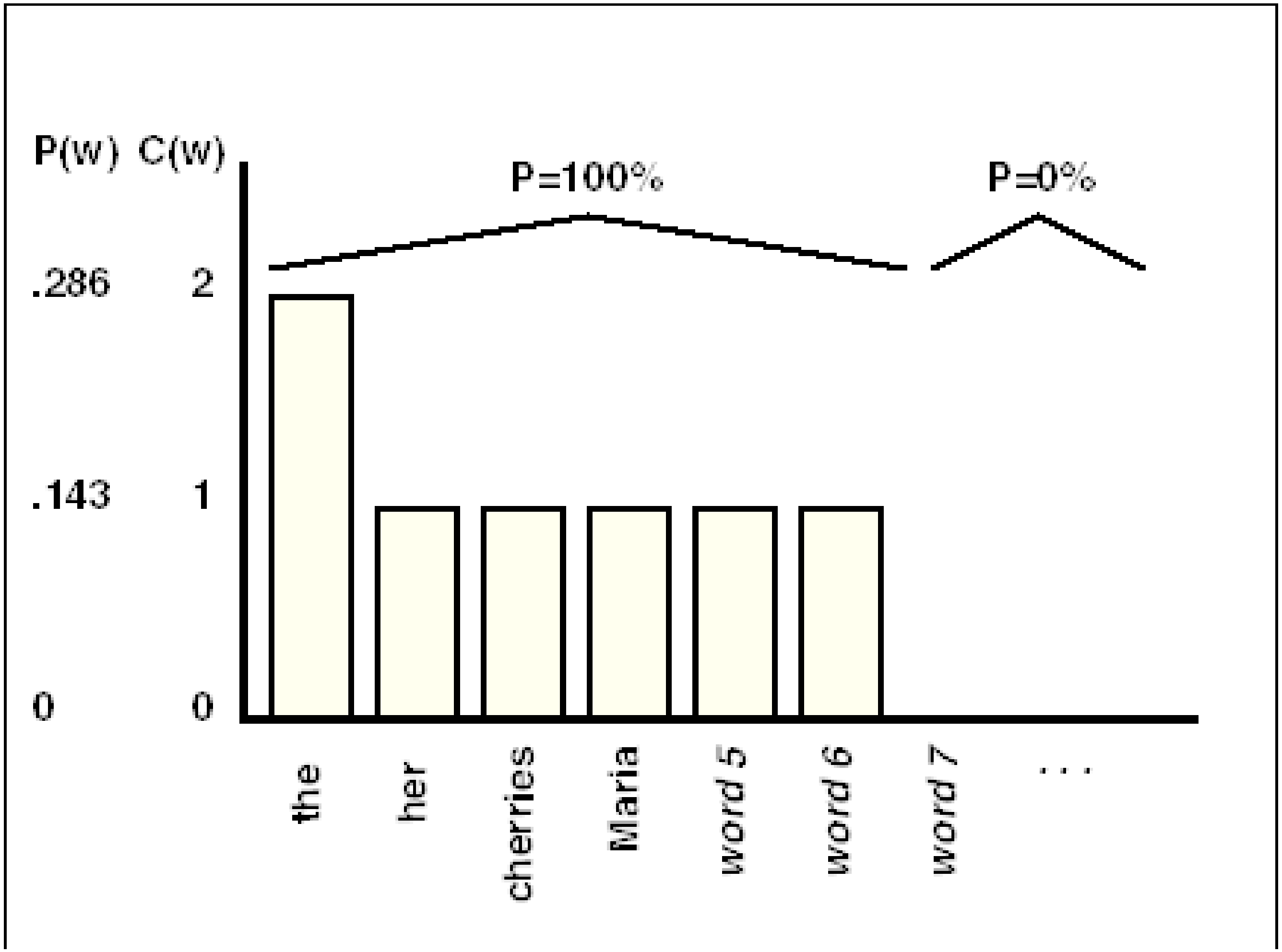
- 2-gram Model

$$P_{MLE}(w_i | w_{i-1}) = \frac{C(w_{i-1} w_i)}{C(w_{i-1})}$$

- 3-gram Model

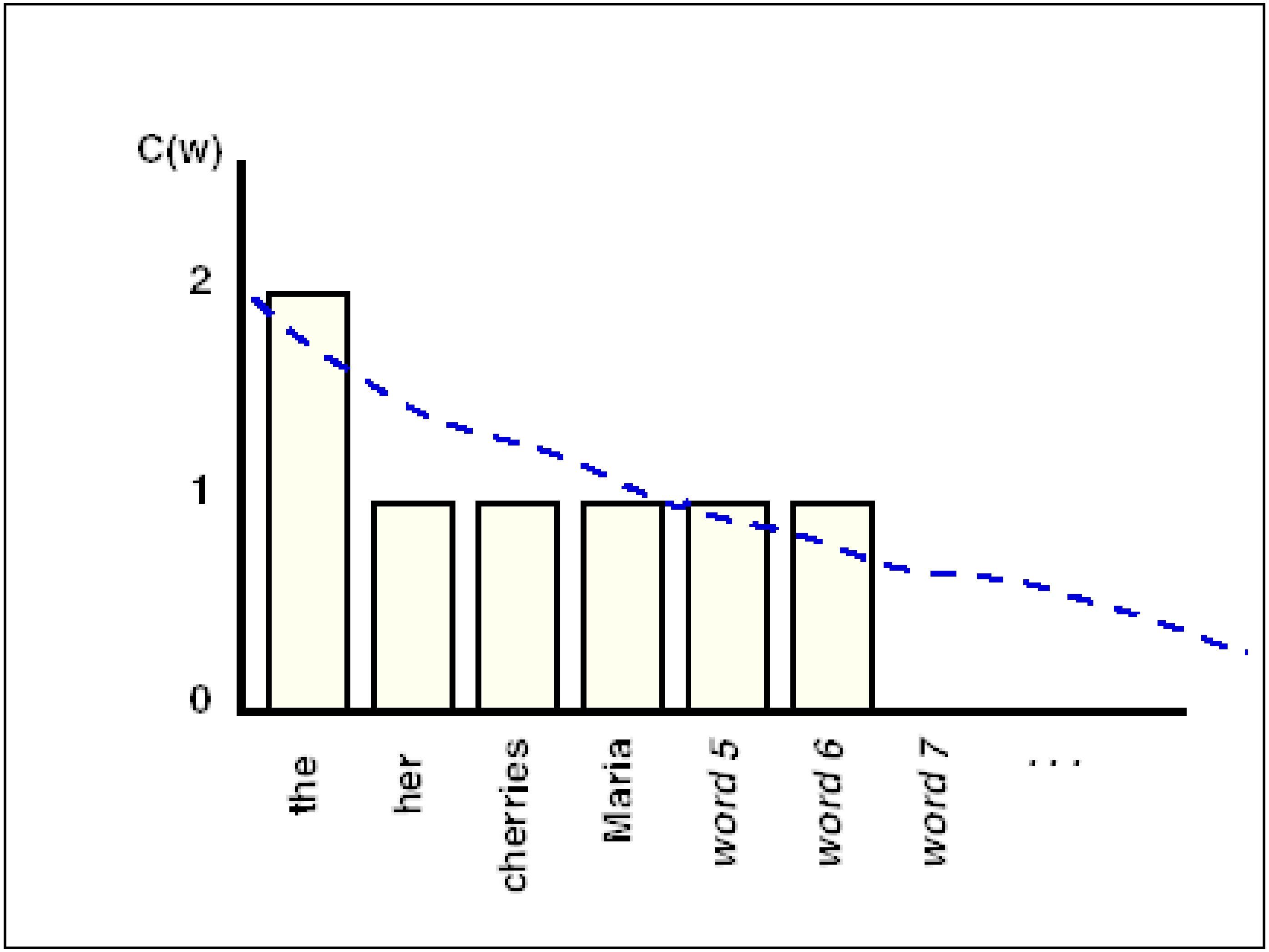
$$P_{MLE}(w_i | w_{i-1}, w_{i-2}) = \frac{C(w_{i-2} w_{i-1} w_i)}{C(w_{i-2} w_{i-1})}$$


C(w)
2
1
0
the
her
cherries
Maria
word 5
word 6
word 7
. . .

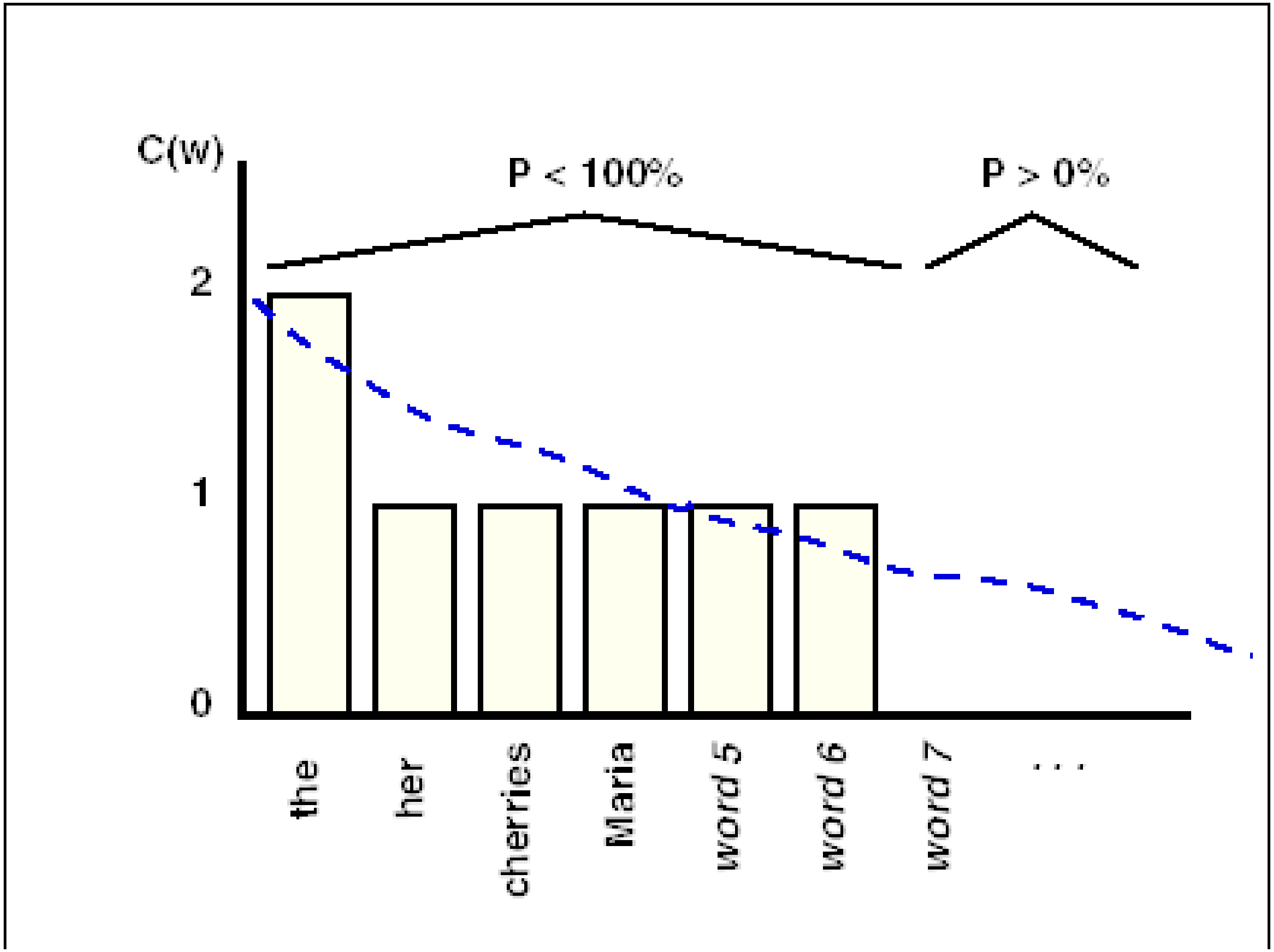


P(w) C(w)
P=100%
P=0%
.286
2
.143
1
0
0
the
her
cherries
Maria
word 5
word 6
word 7
. . .

# True probability distribution



The seen cases are overestimated the unseen ones have a null probability



Save a part of the mass probability from seen cases and assign it to the unseen ones

SMOOTHING

- Some methods perform on the countings:
  - Laplace, Lidstone, Jeffreys-Perks
- Some methods perform on the probabilities:
  - Held-Out
  - Good-Turing
  - Descuento
- Some methods combine models
  - Linear interpolation
  - Back Off