Statistical Language Models

- Language Models, LM
- Noisy Channel model
- Simple Markov Models
- Smoothing

Two Main Approaches to NLP

Knowlege (AI)

- Statistical models

  - Inspired in speech recognition :

probability of next word based on previous

  - Others statistical models

Probability Theory

- X be uncertain outcome of some event.

Called 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

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

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 $w_{1,n}$

Probability Theory

Useful generalizations of the Bayes's law

- To find the probability of something happening calculate the probability that it hapens given some second event times the probability of the second event

- $P(w,x|y,z) = P(w,x)\ P(y,z \mid w,x) \ /\ P(y,z)$

Where x,y,y,z are separates events (i.e. take a word)

- $P(w_1,w_2,..,w_n) = P(w_1)P(w_2|w_1)P(w_3|w_1,w_2),\ldots\ P(w_n|w_1..,w_{n-1})$

also when conditioned on some event

$P(w_1,w_2,..,w_n|x) = P(w_1|x)P(w_2|w_1,x)\ \ldots\ P(w_n|w_1..,w_{n-1},x)$

Statistical Model of a Language

- Statistical models of words of sentences – language models

- Probability of all possible sequences of words .

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

- To create the model use training text and record pairs and triples of words that appear in the text and how many times
  $\mathbf{P(w_i|w_{i-2},w_{i-1}) = C(w_{i-2,i}) / C(w_{i-2,i-1})}$

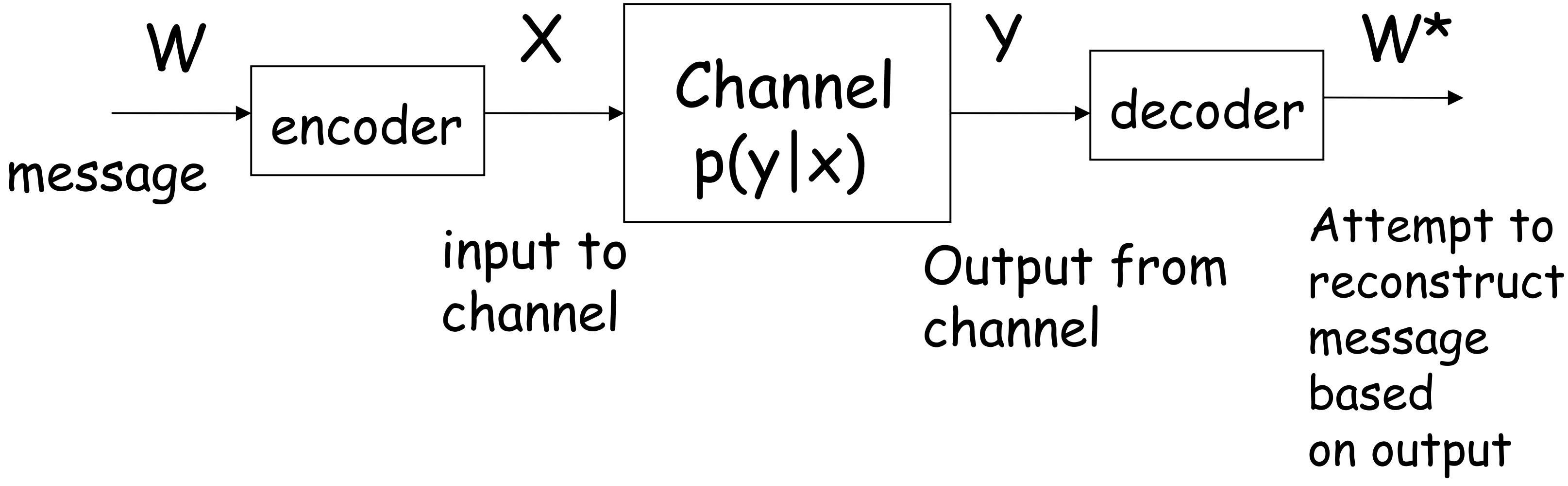
P(submarine|the, yellow) = C(the,yellow, submarine)/C(the,yellow)

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

Language Models

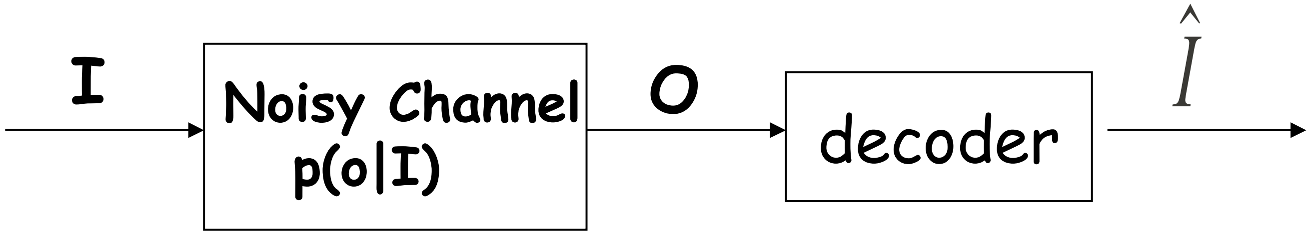
- Statistical Models
- Language Models (LM)
- Vocabulary (V), word
  - $w \in V$
- Language (L), sentence
  - $s \in L$
  - $L \subset V^*$ usually infinite
- $s = w_1, \ldots w_N$
- Probability of s
  - $P(s)$

Noisy Channel Model

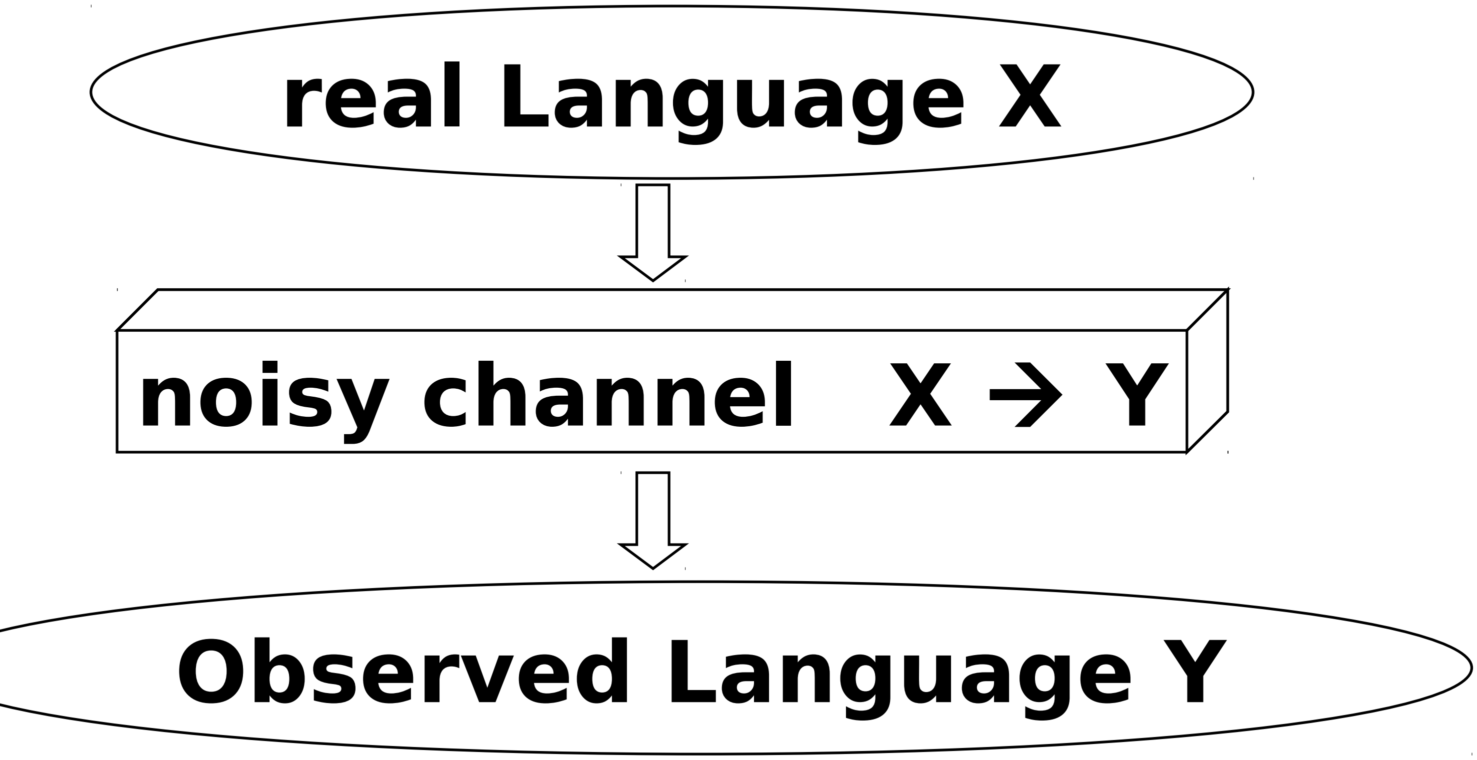


Noisy Channel Model in NLP

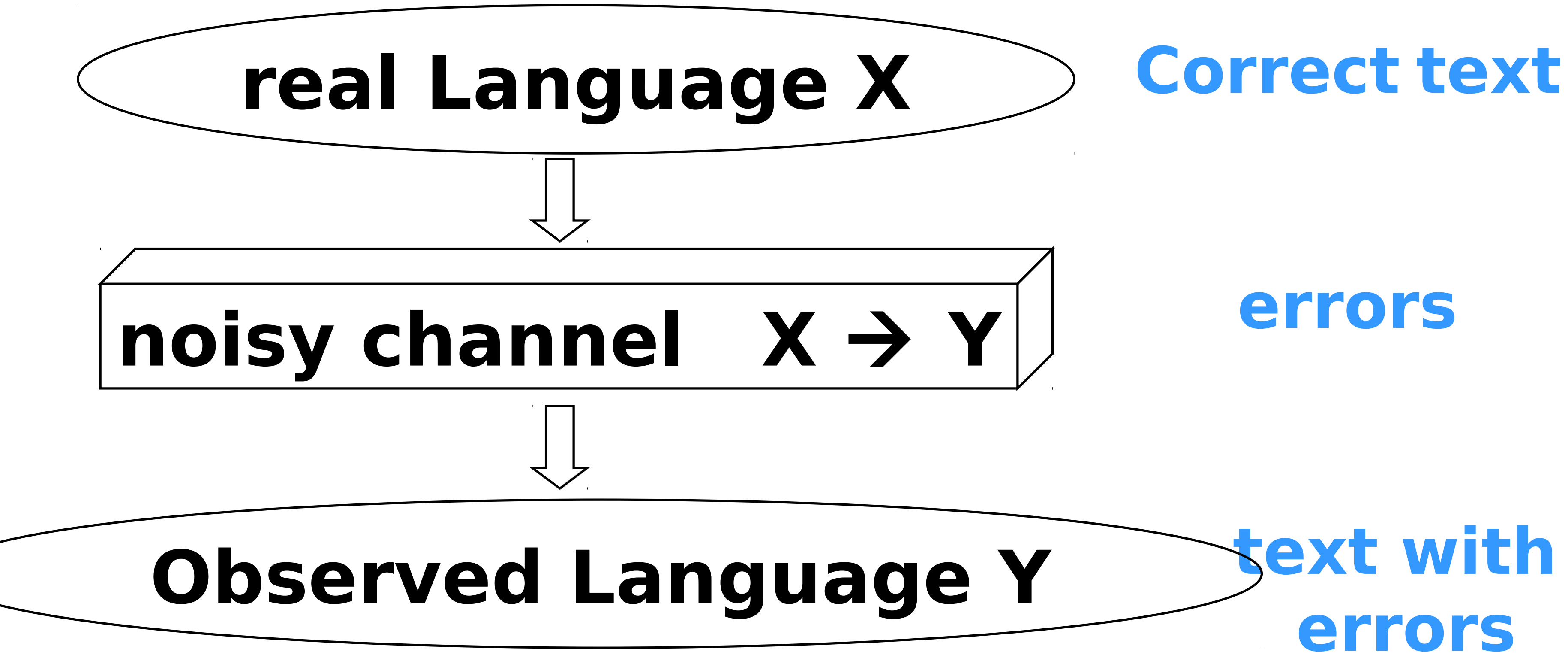
In NLP we do not usually act on coding. The problem is reduced to decode for getting the most likely input given the output,

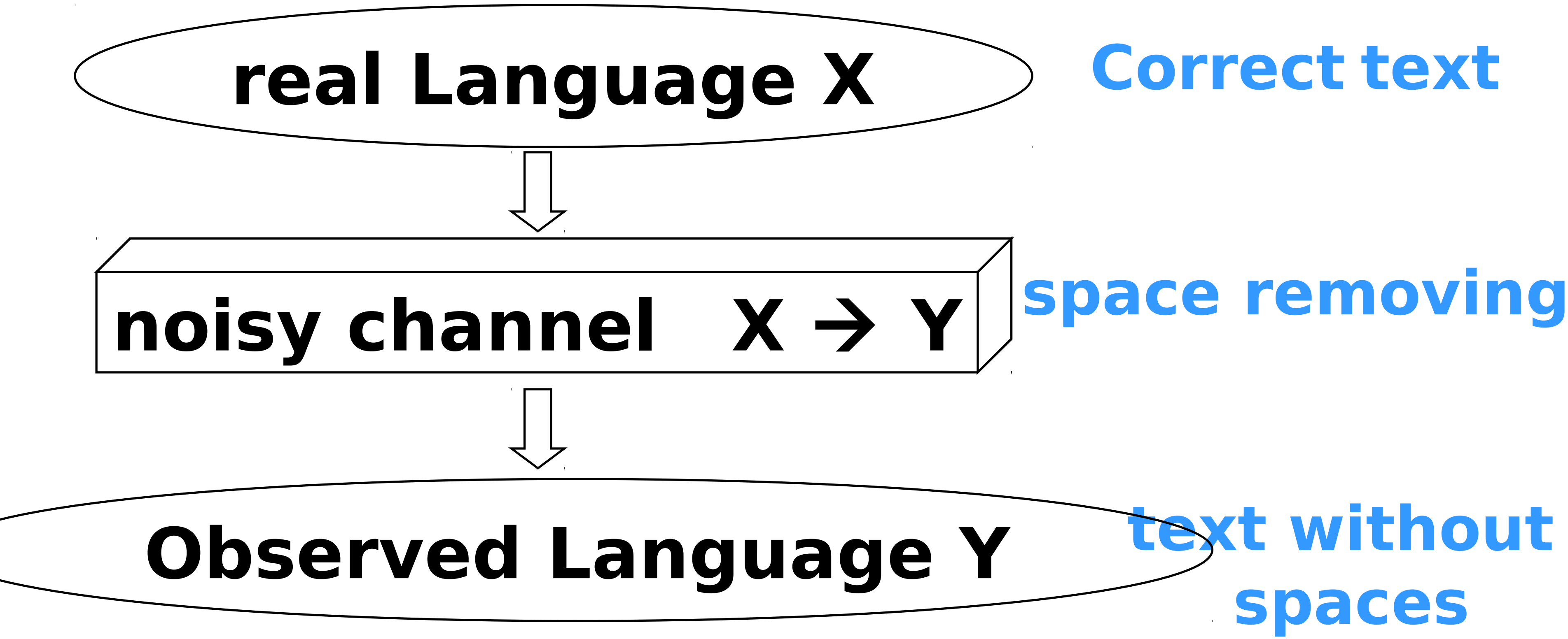


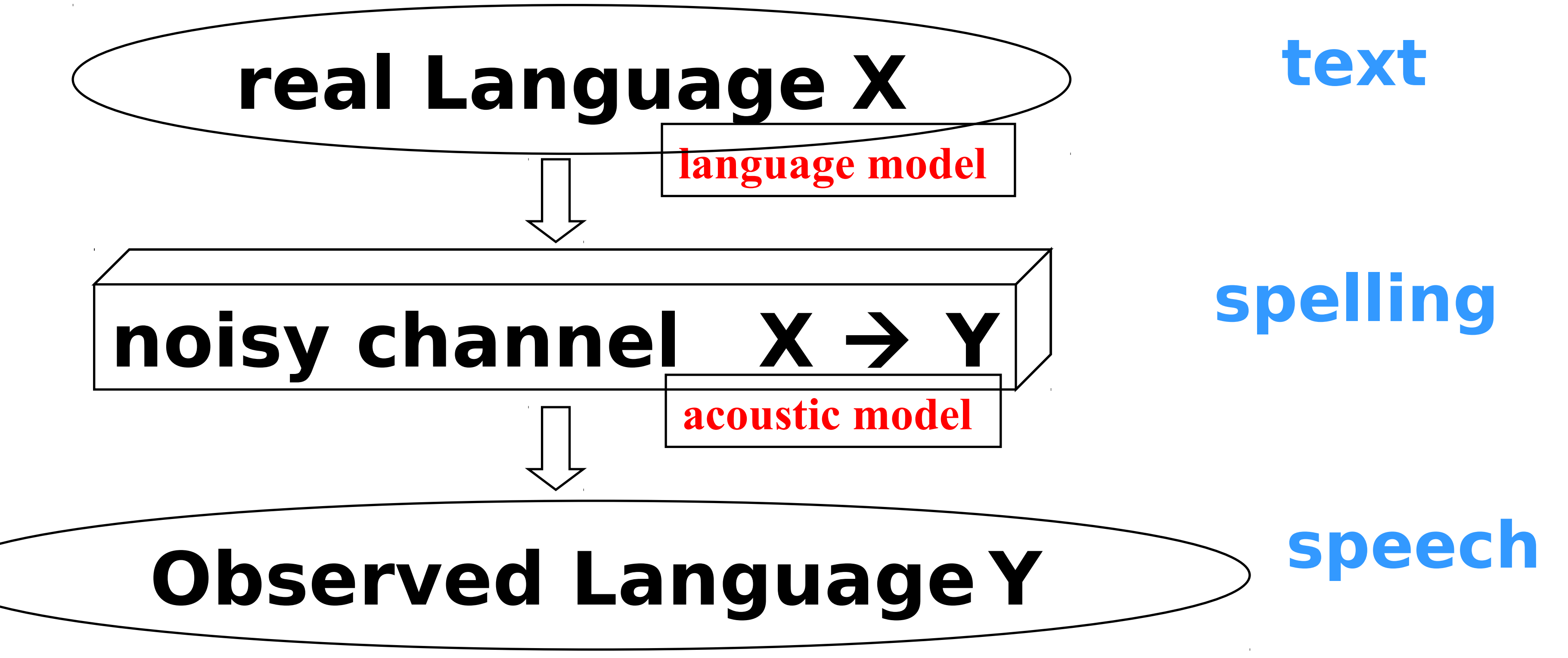
$$p(i)\,p(o|i)$$

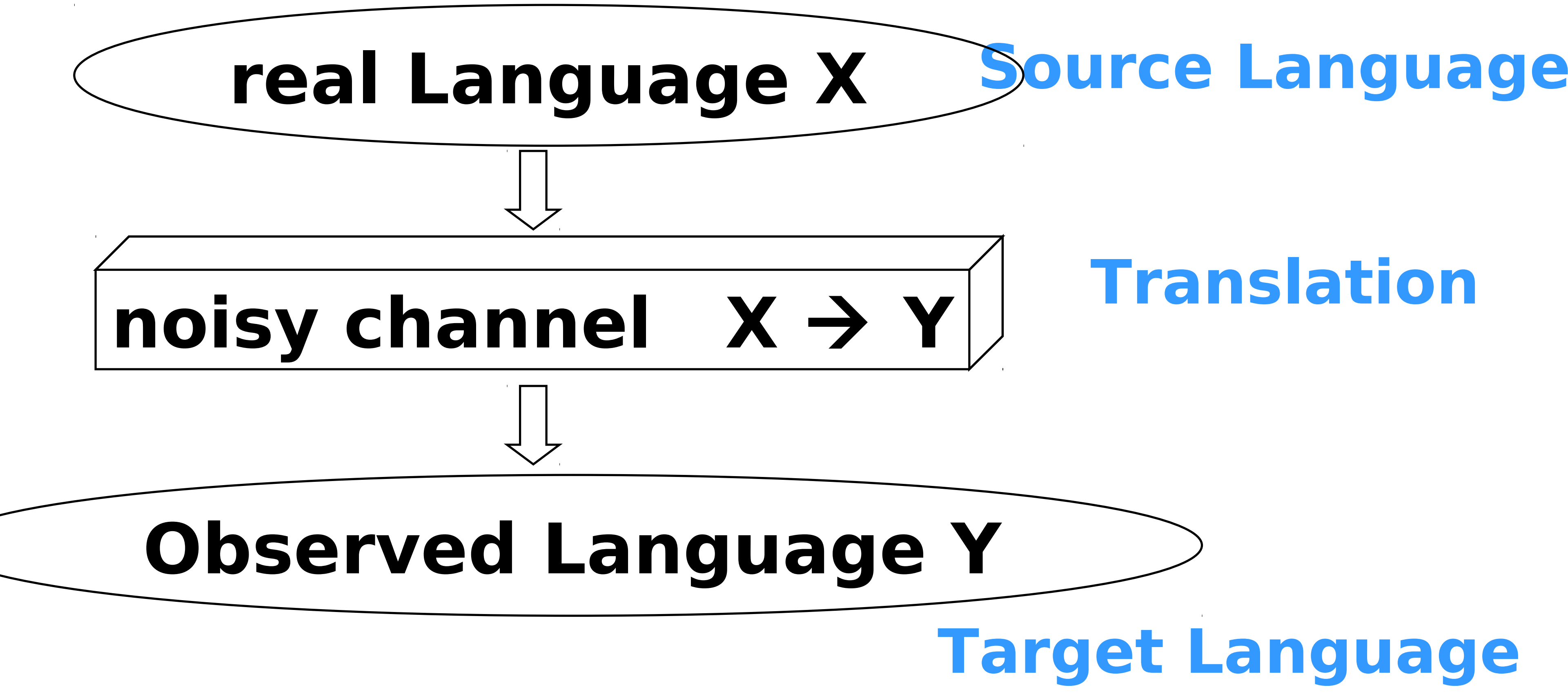


We want to retrieve X from Y


real Language X
Correct text
noisy channel X → Y
errors
Observed Language Y
text with
errors


real Language X
Correct text
noisy channel X → Y
space removing
Observed Language Y
text without spaces


real Language X
language model
noisy channel X → Y
acoustic model
Observed Language Y
text
spelling
speech


real Language X
Source Language
noisy channel X → Y
Translation
Observed Language Y
Target Language

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

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

- History
  - $h_i = \{ w_i, \dots w_{i-1} \}$
- Markov Models

- 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

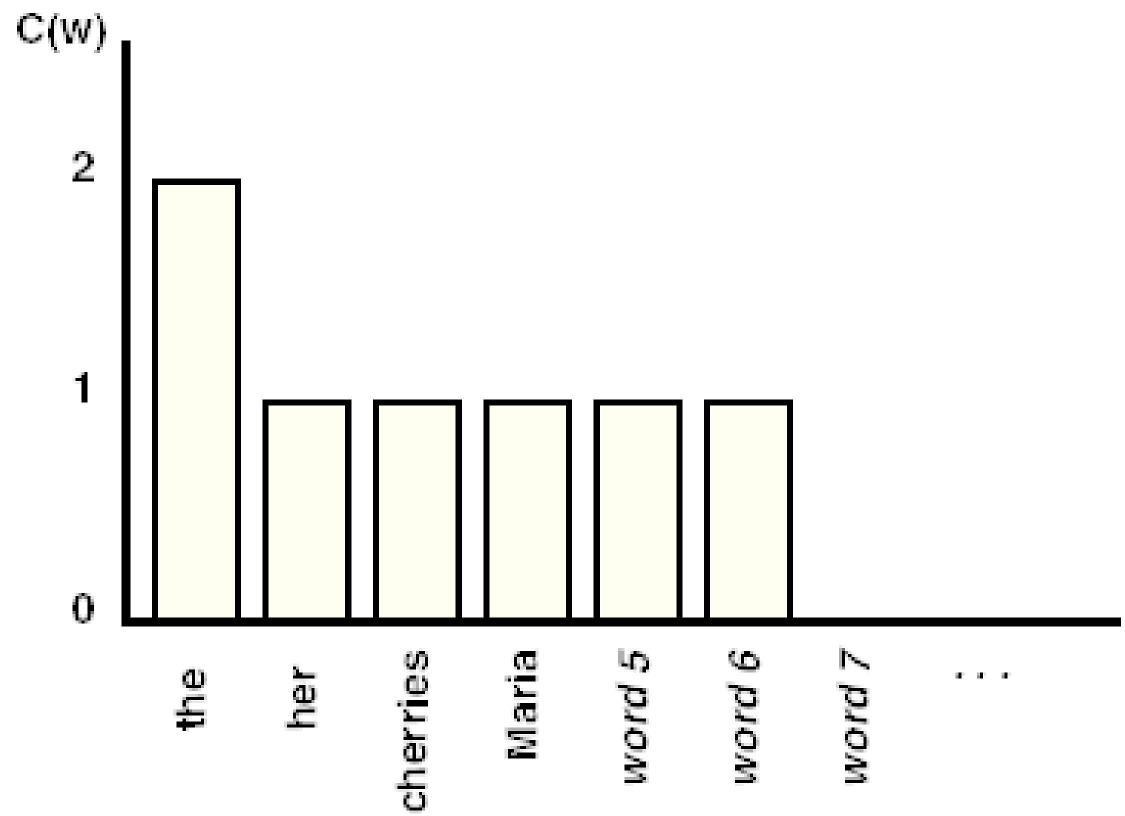
$$P_{MLE}(w) = \frac{C(w)}{|V|}$$

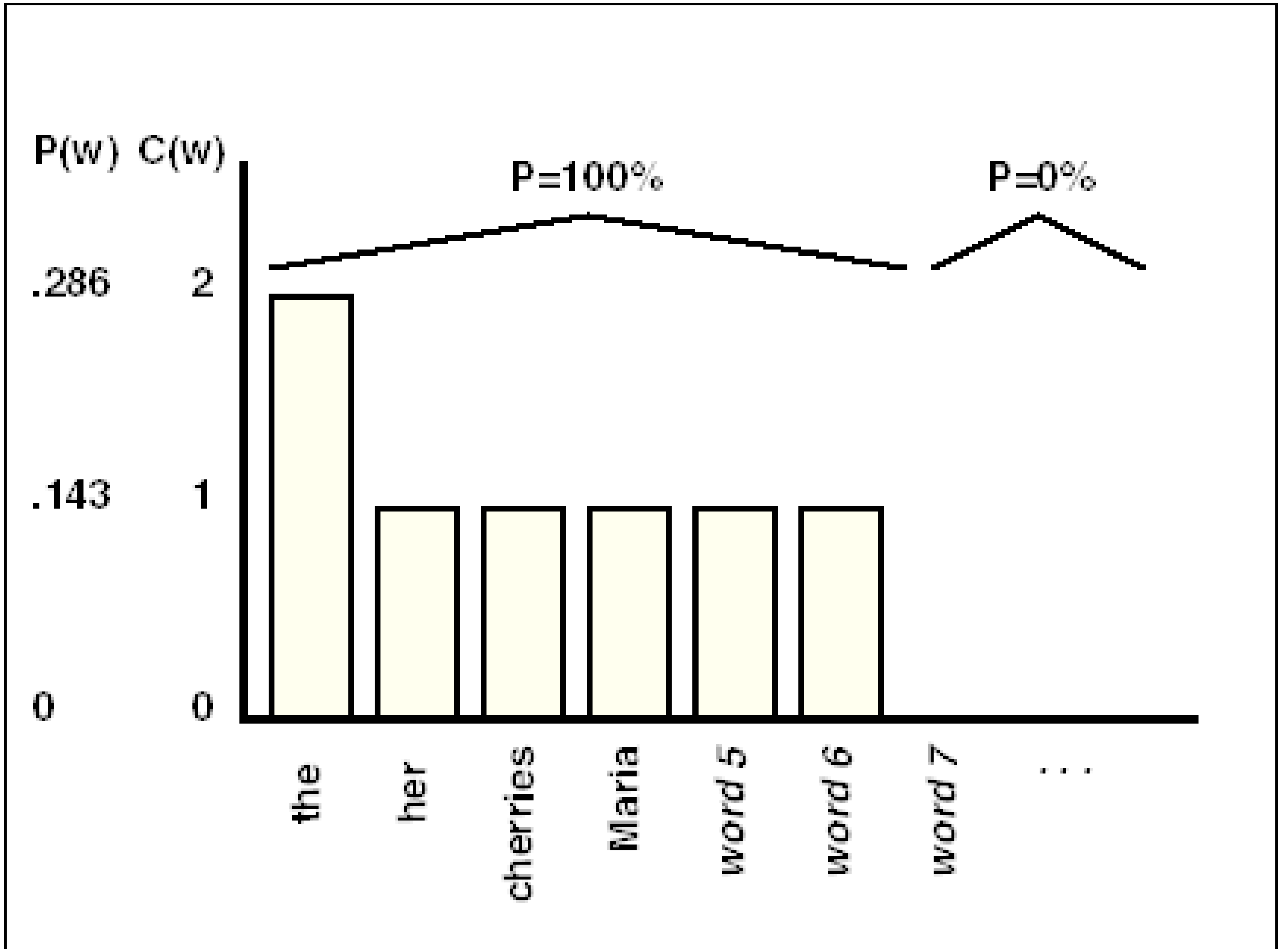
- 2-gram Model

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

- 3-gram Model

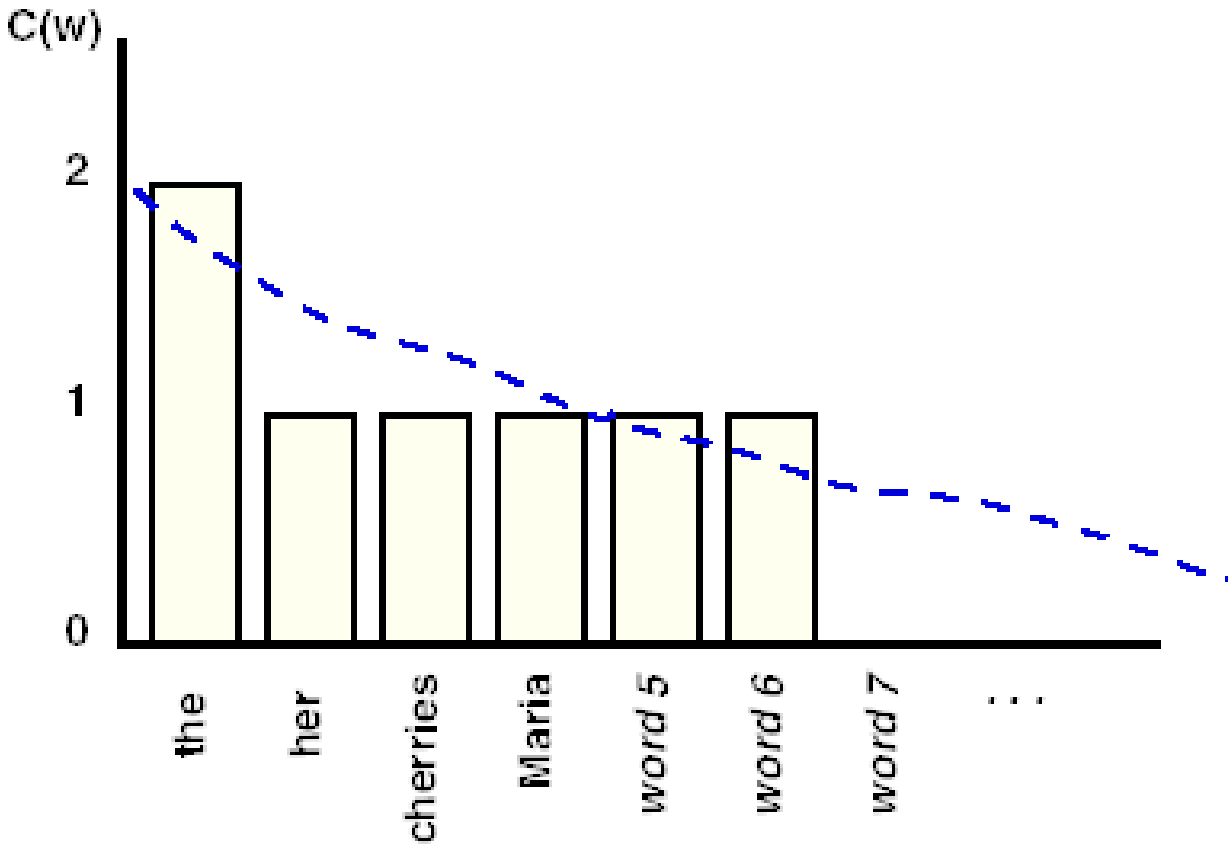
$$P_{MLE}(w_i | w_{i-1}, w_{i-2}) = \frac{C(w_{i-2} w_{i-1} w_i)}{C(w_{i-2} w_{i-1})}$$


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

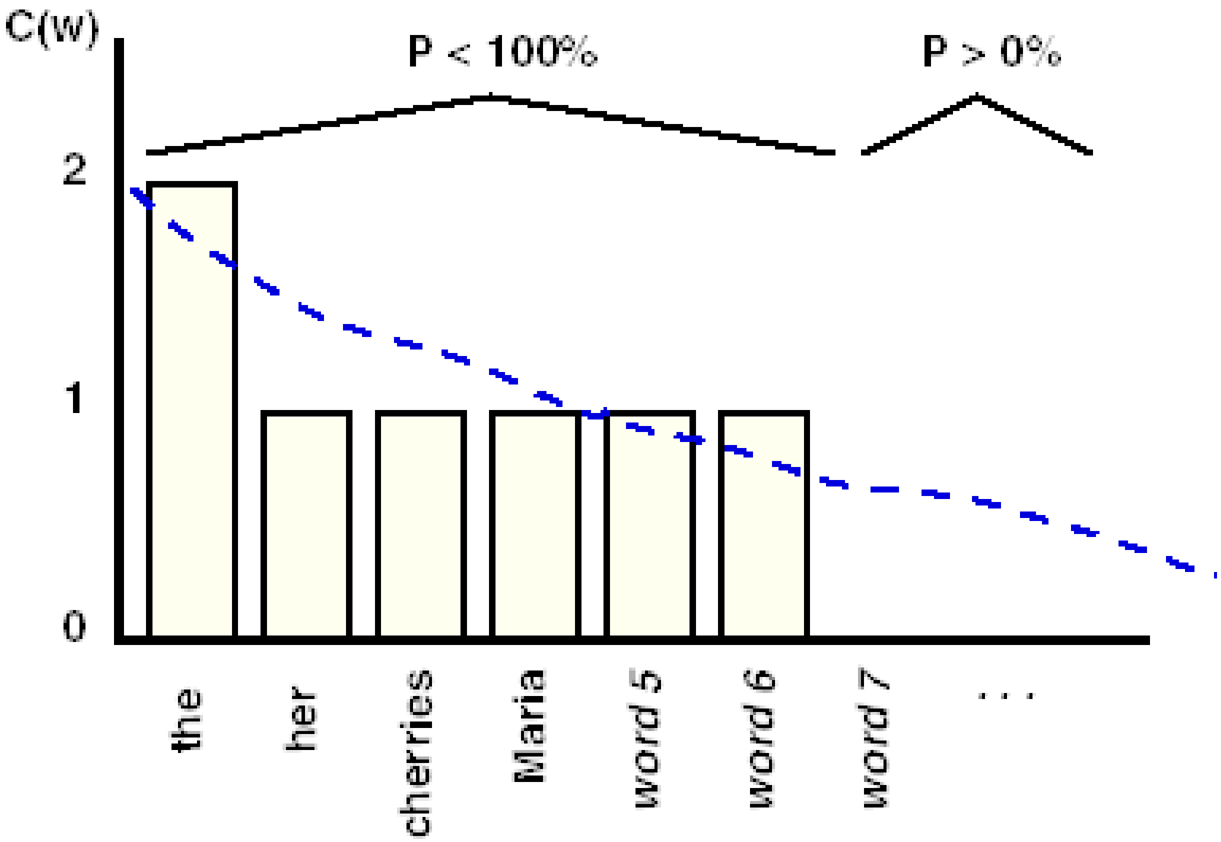


P(w)
C(w)
P=100%
P=0%
.286
2
.143
1
0
0
the
her
cherries
Maria
word 5
word 6
word 7
. . .

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

Laplace (add 1)

$$P_{laplace}(w_1 \cdots w_n) = \frac{C(w_1 \cdots w_n) + 1}{N + B}$$

P = probability of an n-gram

C = counting of the n-gram in the training corpus

N = total of n-grams in the training corpus

B = parameters of the model (possible n-grams)

Lidstone (generalization of Laplace)

$$P_{Lid}(w_1 \cdots w_n) = \frac{C(w_1 \cdots w_n) + \lambda}{N + B \cdot \lambda}$$

$\lambda =$ small positive number

M.L.E: $\lambda = 0$
Laplace: $\lambda = 1$
Jeffreys-Perks: $\lambda = ½$

Held-Out

- Compute the percentage of the probability mass that has to be reserved for the n-grams unseen in the training corpus
- We separate from the training corpus a held-out corpus
- We compute howmany n-grams unseen in the training corpus occur in the held-out corpus
- An alternative of using a held-out corpus is using Cross-Validation
  - Held-out interpolation
  - Deleted interpolation

Held-Out

Let a n-gram $w_1 \dots w_n$

$r = C(w_1 \dots w_n)$

$C_1(w_1 \dots w_n)$ counting of the n-gram in the training set

$C_2(w_1 \dots w_n)$ counting of the n-gram in the held-out set

N_r number of n-grams with counting r in the training set

$$T_r = \sum_{\{w_1 \cdots w_n : C_1(w_1 \cdots w_n) = r\}} C_2(w_1 \cdots w_n)$$

$$P_{ho}(w_1 \cdots w_n) = \frac{T_r}{N_r N}$$

Good-Turing

$$r = (r+1)\frac{E(N_{r+1})}{E(N_r)} \qquad\qquad P_{GT} = r/N$$

r^* = adjusted count

N_r = number of *n-gram*-types occurring *r* times

$E(N_r)$ = expected value

$E(N_{r+1}) < E(N_r)$

Zipf law

Combination of models: Linear combination (interpolation)

$$P_{li}(w_n|w_{n-2},w_{n-1})=$$

$$\lambda_1 P_1(w_n)+\lambda_2 P_2(w_n|w_{n-1})+\lambda_1 P_3(w_n|w_{n-2},w_{n-1})$$

- Linear combination of de 1-gram, 2-gram, 3-gram, ...
- Estimation of λ using a development corpus

Katz’s Backing-Off

- Start with a n-gram model
- Back off to n-1 gram for null (or low) counts
- Proceed recursively

- Performing on the history

$$h_i = \Phi(w_1^{i-1})$$

- Class-based Models
  - Clustering (or classifying) words into classes
    - POS, syntactic, semantic
  - Rosenfeld, 2000:
    - P(wi|wi-2,wi-1) = P(wi|Ci) P(Ci|wi-2,wi-1)
    - P(wi|wi-2,wi-1) = P(wi|Ci) P(Ci|wi-2,Ci-1)
    - P(wi|wi-2,wi-1) = P(wi|Ci) P(Ci|Ci-2,Ci-1)
    - P(wi|wi-2,wi-1) = P(wi|Ci-2,Ci-1)

- Structured Language Models
  - Jelinek, Chelba, 1999
  - Including the syntactic structure into the history

$$P(w_i \mid h_i) = \sum_{T_i} P(w_i, T_i \mid w_1^{i-1})$$

- T_i are the syntactic structures
  - binarized lexicalized trees