

# Today's Topic:

## Language Modeling Part 2

### Reference:

- Review paper from Roni Rosenfeld, IEEE 2000
- Huang et al. Spoken Language Processing, Chapter 11
- Jerome Bellegarda, Speech Communication 2004
- Jelinek, Statistical Methods for Speech Recognition, MIT Press, Cambridge, MA, 1997

# Motivation

- N-gram LM cannot capture long-range dependency
  - e.g. trigram only considers previous 2 words
  - Long-distance context provides useful information
- e.g. I booked a flight to **Germany**. The flight will first arrive at (**Frankfurt**? Or Pittsburgh?) international airport...
- e.g. **HMM** is a powerful model, which can be applied to various applications such as **speech recognition**, ...

# Other Kinds of Language Models

- Cache language models  
(constantly adapting to a floating text)
- Trigger language models  
(can handle long distance effects)
- Context free grammar language models  
(use simple and efficient LM-definition)
- Decision tree language models  
(handle long distance effects, use rules)
- HMM language models  
(stochastic decision for combination of independent LMs)
- Maxent LM  
(feature integrations)
- LSA-based LM  
(exploit topical context)

# Cache Language Models

Observation: When using a speech recognizer (e.g. for dictating news texts) the topics can change at arbitrary points.

- **Idea:**

Use a static and a dynamic component of the language model.  
Constantly update the dynamic component.

- **Static Component:**

$P_S(w_k | w_{k-(n-1)} \dots w_{k-1})$  i.e. the usual trained language model.

- **Dynamic Component:**

complete  $n$ -gram language model constructed from the text dictated so far,  
e.g.:

$$P_{\text{Cache}}(w_k | w_{k-(n-1)} \dots w_{k-1}) = 0.5 \cdot f(w_k) + 0.25 \cdot f(w_k | w_{k-1}) + 0.25 \cdot f(w_k | w_{k-2} w_{k-1})$$

- **Total Language Model:**

$$P_T = \lambda_{\text{Cache}} \cdot P_{\text{Cache}} + (1 - \lambda_{\text{Cache}}) \cdot P_S$$

- **Variation:**

- Compute  $P_{\text{Cache}}$  on window of last  $l$  words.
- Vary  $\lambda_{\text{Cache}}$  with the size of the cache

# Trigger Language Models

Observation: Often, the probability of a word depends on some words far in the history. Often, when a content-carrying word occurs in a text it is likely to occur again some time later.

- **Idea:**

Use a standard interpolated  $n$ -gram language model.

But constantly update the unigrams  $P(w_k)$  for some words  $w_k$ .

- **Trigger**

For each word  $w$  define a trigger list  $L(w)$  (possibly weighted) of words that are likely to occur some time later.

E.g.  $L(\text{MONEY}) = \{\text{BANK, SAVINGS, ACCOUNT, DOLLARS, COST}\}$ .

Then, for each word  $v$  in  $L(w)$  increase  $P(v)$  by some value.

# Context-Free Grammar Language Models

Why use context-free grammars (CFG) instead of  $n$ -grams?

- The rules of the grammar can be written by expert without the need for lots of training text data.
- CFGs are powerful enough to represent large parts of natural languages.
- CFGs are constrained enough to allow efficient search space reduction.
- If interpreted as finite state automaton, then the state transition sequence can also be used for efficient parsing (semantic analysis) of the sentence.

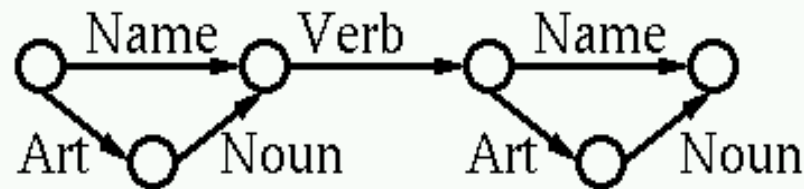
S := NP VP

VP := Verb NP

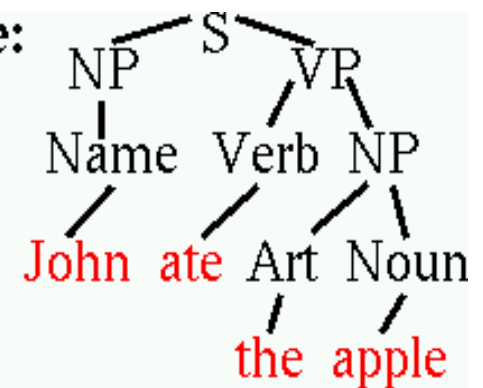
NP := Name

NP := Art Noun

**Transition Network:**



**Parse Tree:**



# Tree-Based Language Models

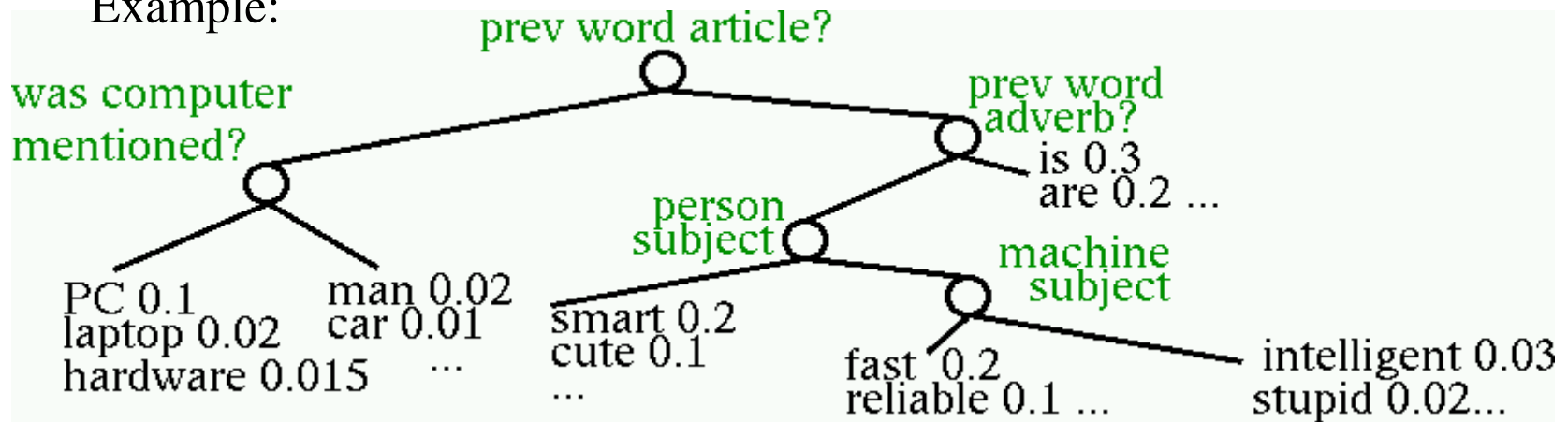
## Observation:

- $n$ -gram models have very limited context
- Parameters are almost un-trainable for large values of  $n$

## CART-based Technique: (Classification and Regression Trees)

- Each tree node asks question about previous  $k$  words ( $k$  is approximately 20)
- Each leaf node is a probability distribution on vocabulary (unigrams)
- The tree is automatically generated (including questions)
- Criterion is to minimize leaf node entropy.
- Compare to clustering of acoustic context-dependent models.

Example:



# Hidden Markov Models for Language Modeling

## Observation:

- Often, spoken speech is carried out in phases.  
Conversation: greeting phase, small-talk phase, good-bye  
Dictation: Writing a letter: addresses, opening, content, closing
- The typical word sequences in one phase differ from those in another phase.
- We can regard the conversation phases as states of the conversation.  
Transitions from one state into another can be regarded as a stochastic process.

## Approach:

- Build / train HMM that represents conversation phases and transitions, e.g.:



- The states emit language models, or mixture weight distributions (i.e. interpolation factors for different specialized language models).

# Maximum Entropy (Maxent) LM

- Motivation: treat LM training as constraint optimization
  - First satisfy all constraints
  - Then seek the most unbiased / “uniform” model
- e.g: Maxent training of a die with 6 faces
  - No constraint  $\Rightarrow \text{Pr}(\text{“1”}) = \dots = \text{Pr}(\text{“6”}) = 1/6$
  - If Oracle says:  $\text{Pr}(\text{“1”}) = 0.3$   
 $\Rightarrow \text{Pr}(\text{“2”}) = \text{Pr}(\text{“3”}) \dots = \text{Pr}(\text{“6”}) = (1-0.3) / 5$

# Exponential form

- knowledge integration as features in  $\exp(\cdot)$  model:

Binary feature function (0/1)

$$P_{me}(w|h) = \frac{e^{\sum_j^J \lambda_j \cdot f_j(w,h)}}{Z(h)}$$

Feature weights (model parameters)

# Sample binary feature functions

A feature is activated (set to 1) if an event (w,h) occurs

e.g. word-trigger feature:

$$f_{\langle HMM, speech \rangle}(w, h) = \begin{cases} 1 & \text{if } w = \text{"HMM"} \text{ and } h \text{ contains "speech"} \\ 0 & \text{otherwise} \end{cases}$$

e.g. POS feature:

$$f_{\langle NOUN, DET \rangle}(w, h) = \begin{cases} 1 & \text{if } POS(w) = \text{"NOUN"} \text{ and } POS(w_{-1}) = \text{"DET"} \\ 0 & \text{otherwise} \end{cases}$$

# N-gram feature functions

- Word unigram feature **template**: (u is a word in vocabulary)
  - How many unigram features are there?

$$f_{\langle u \rangle}(w, h) = \begin{cases} 1 & \text{if } w = "u" \\ 0 & \text{otherwise} \end{cases}$$

- Word bigram feature template:

$$f_{\langle u, v \rangle}(w, h) = \begin{cases} 1 & \text{if } w_{i-1} = "u" \text{ and } w = "v" \\ 0 & \text{otherwise} \end{cases}$$

# Maxent principle

Maximize the entropy of the LM such that **expected feature counts (model) = empirical feature counts**:

$$\begin{aligned} E_P[f_j(w, h); \{\lambda_j\}] &= E_{\tilde{P}}[f_j(w, h)] \\ &= \frac{\sum_{i=1}^N f_j(w_i, h_i)}{N} \end{aligned}$$

Model expectation

Avg. feature counts  
(N= # of training tokens)

⇔ MLE on exponential model (Primal-Dual)

# Maxent training

- Gradient ascent
  - Compute the gradient of training log likelihood w.r.t each lambda
  - Perform gradient ascent
- Generalized iterative scaling (GIS)
  - Each iter guarantees likelihood improvement

$$\lambda_j^{(new)} = \lambda_j^{(old)} + \frac{1}{C} \cdot \log \frac{E_{\tilde{P}}[f_j(w, h)]}{E_P[f_j(w, h); \{\lambda_j^{(old)}\}]}$$

Max. feature counts

# Pros & Cons

## Pros:

- Flexible for knowledge integration (feature engineering)
- Globally optimal solution (convex problem)

## Cons:

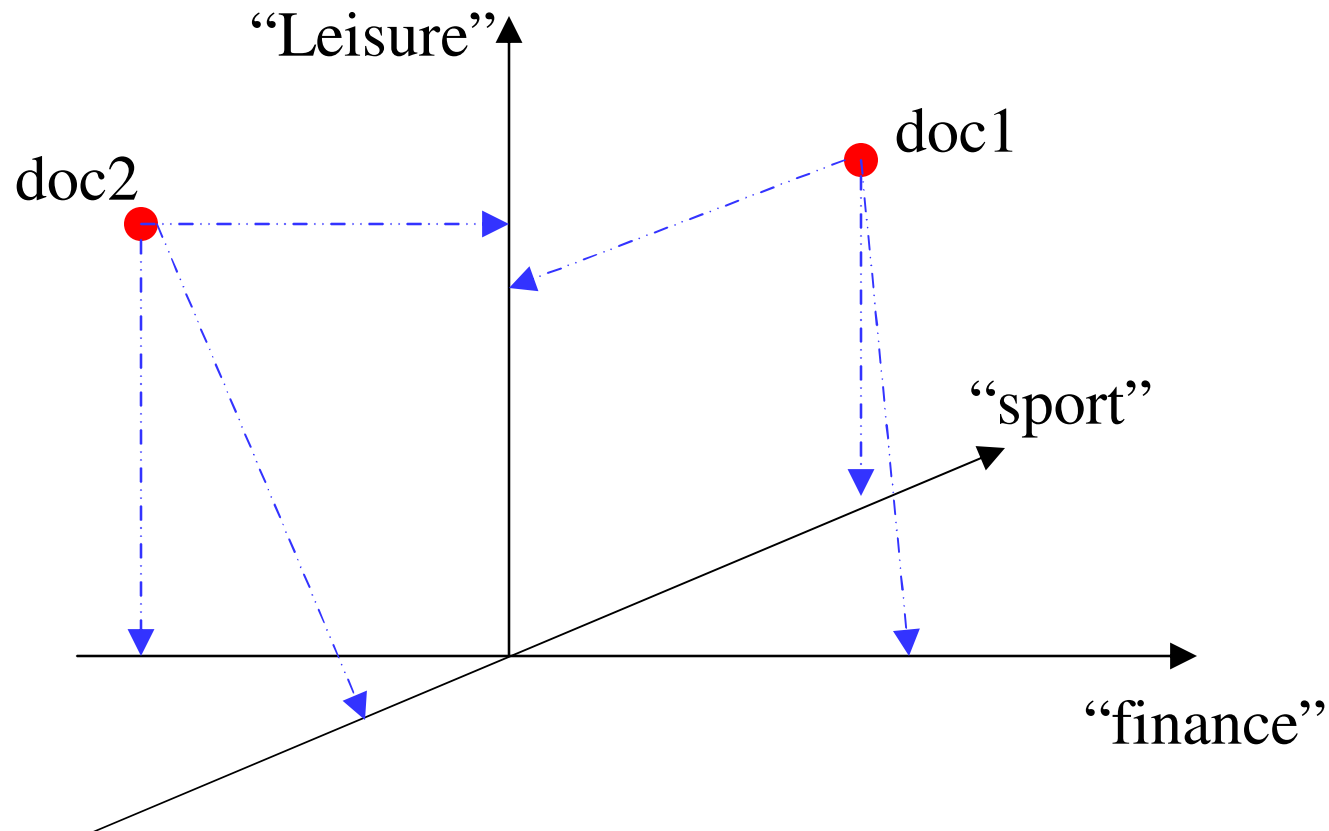
- very expensive to train (iterative procedure)
- Expensive computation of  $Z(h)$

# Latent Semantic Analysis (LSA)

- Motivation: Capture latent topics in a document (unsupervised)
  - “eigenvectors”
- Latent Semantic Indexing (LSI)
  - Vector-space model
  - Singular-value decomposition
- Probabilistic LSI (pLSI)
  - Probabilistic model
  - EM algorithm

# LSI

- Project document into different “semantic” dimension
- Dimensionality reduction:  $V \rightarrow k$
- Coordinates can be negative!



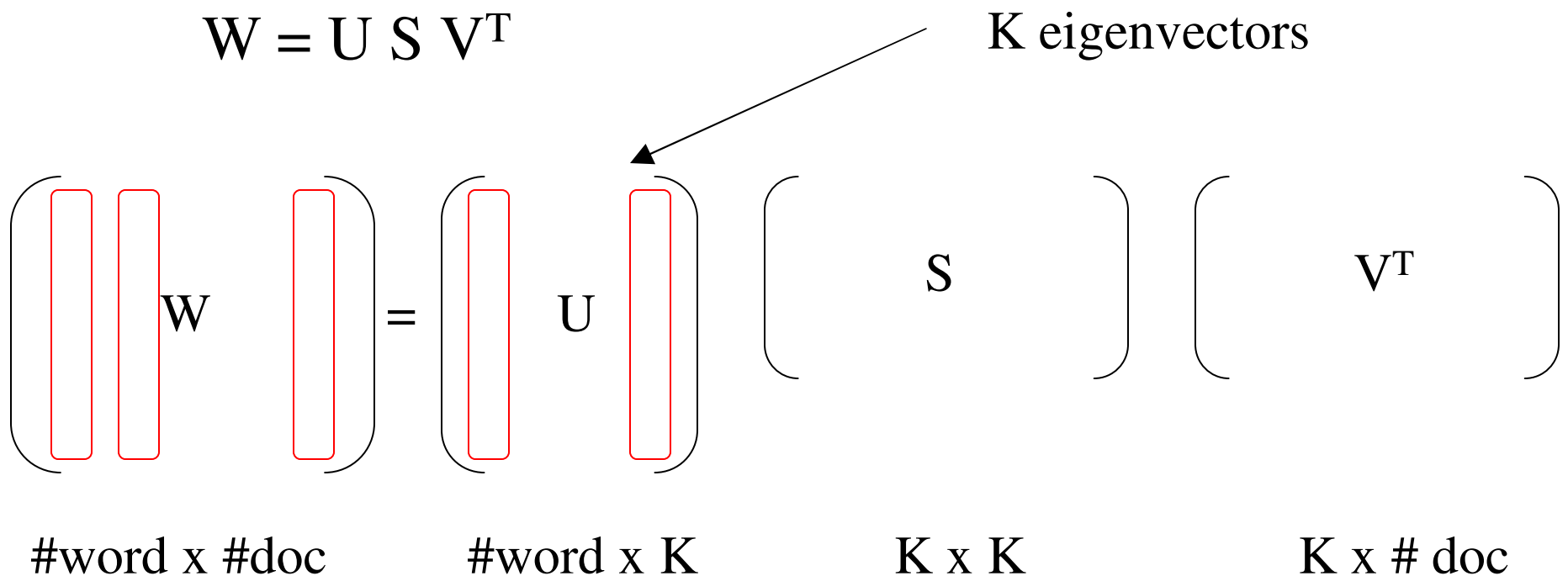
# Singular value decomposition

- Find topical eigenvectors via SVD

Steps:

1. Prepare term-document matrix  $W$
2. Decompose  $W$ :

$$W = U S V^T$$



## LSI-based LM [Bellegarda 1997]

Define similarity between current word  $w$  & history  $h$ :

$$P_{LSI}(w|h) = \frac{sim(w, h)^\gamma}{\sum_v sim(v, h)^\gamma}$$

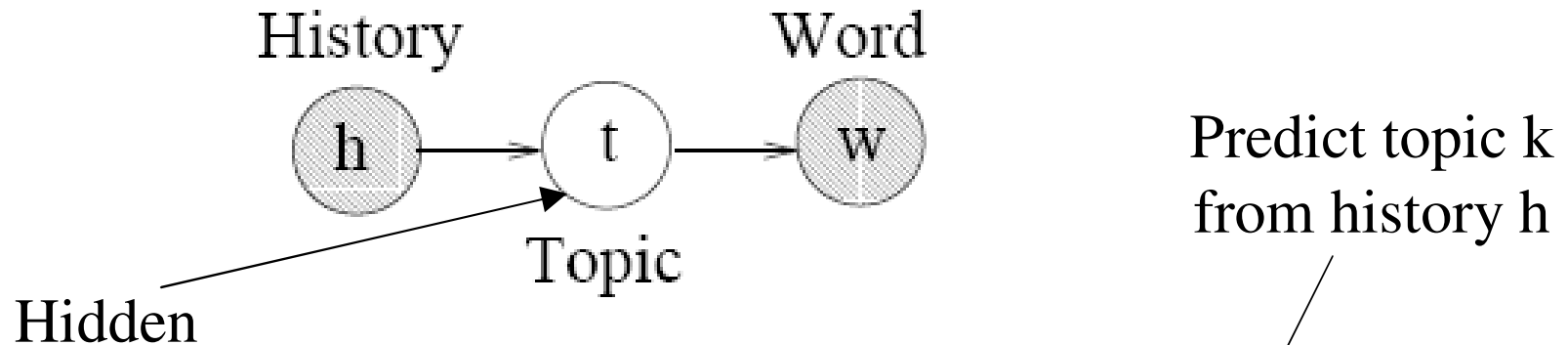
To compute  $sim(w, h)$ :

1. project  $w$  into LSA space (output dim =  $K$ )
2. project  $h$  into same LSA space (output dim =  $K$ )
3. Perform cosine similarity between projected vectors

# pLSI [Gildea & Hofmann 1999]

Move from vector-space model to probabilistic model

- Topic mixture model



$$P_{pLSI}(w|h) = \sum_k P(w|k) \cdot P(k|h)$$

Predict w given topic k

# pLSI training

Hidden topic label for  $w \Rightarrow$  EM training algorithm

**E-step:** (determine topic posterior for each word  $i$  in context  $h$ )

$$P^{(new)}(t_i|h) \propto P^{(old)}(w_i|t_i) \cdot P^{(old)}(t|h)$$

word-level topic posterior Doc-level topic posterior

**M-step:** determine model parameters:

$$P^{(new)}(t|h) \propto \sum_{i=1}^N P^{(new)}(t_i|h)$$

Intuition: Summing all word-level posteriors

$$P^{(new)}(w|t) \propto \sum_{i=1}^N P^{(new)}(t_i|h) \cdot \delta(w, w_i)$$

Weighted relative frequency

## Summary (Part 2)

- Covered different LM approaches
- Share the same goal: To capture long-range dependence in natural language
  - Cache, Trigger, Tree, HMM, CFG, Maxent, Topic ... etc