DT2118
Speech and Speaker Recognition
Language Modelling

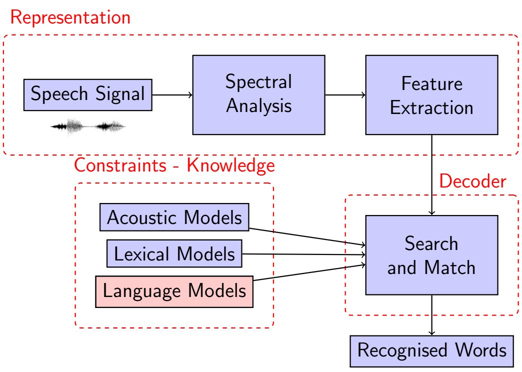
Giampiero Salvi

KTH/CSC/TMH giampi@kth.se

VT 2015

Notes

Components of ASR System



Notes

Why do we need language models?

Bayes' rule:

$$P(\text{words}|\text{sounds}) = \frac{P(\text{sounds}|\text{words})P(\text{words})}{P(\text{sounds})}$$

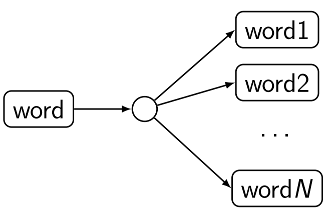
where
$P(\text{words})$: *a priori* probability of the words (Language Model)

We could use non informative priors ($P(\text{words}) = 1/N$), but...

Notes

Branching Factor

- if we have N words in the dictionary
- at every word boundary we have to consider N equally likely alternatives
- N can be in the order of millions



Notes

Ambiguity

"ice cream" vs "I scream"
/aɪ s k ɹ iː m/

Notes

Language Models in ASR

We want to:

1. limit the branching factor in the recognition network
2. augment and complete the acoustic probabilities

- we are only interested to know if the sequence of words is **plausible** grammatically or not
- this kind of grammar is **integrated** in the recognition network **prior to decoding**

Notes

Language Models in Dialogue Systems

- we want to assign a class to each word (noun, verb, attribute... parts of speech)
- parsing is usually performed on the output of a speech recogniser

The grammar is used twice in a Dialogue System!!

Notes

Language Models in ASR

- small vocabulary: often **formal** grammar specified by hand
- example: loop of digits as in the HTK exercise
- large vocabulary: often **stochastic** grammar estimated from data

Notes

Formal Language Theory

grammar: formal specification of permissible structures for the language

parser: algorithm that can analyse a sentence and determine if its structure is compliant with the grammar

Notes

Chomsky's formal grammar

Noam Chomsky: linguist, philosopher, ...

$$G = (V, T, P, S)$$

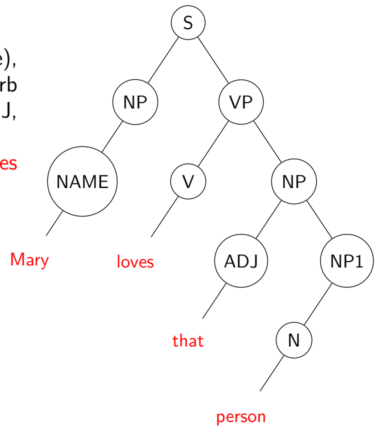
where

V: set of non-terminal constituents
T: set of terminals (lexical items)
P: set of production rules
S: start symbol

Notes

Example

$S =$ sentence
$V =$ {NP (noun phrase), NP1, VP (verb phrase), NAME, ADJ, V (verb), N (noun)}
$T =$ {Mary , person , loves , that , . . . }
$P =$ {S → NP VP
NP → NAME
NP → ADJ NP1
NP1 → N
VP → VERB NP
NAME → Mary
V → loves
N → person
ADJ → that }



Notes

Chomsky's hierarchy

Greek letters: sequence of terminals or non-terminals
Upper-case Latin letters: single non-terminal
Lower-case Latin letters: single terminal

Types	Constraints	Automata
Phrase structure grammar	$\alpha \to \beta$. This is the most general grammar	Turing machine
Context-sensitive grammar	length of $\alpha \leq$ length of β	Linear bounded
Context-free grammar	$A \to \beta$. Equivalent to $A \to w, A \to BC$	Push down
Regular grammar	$A \to w, A \to wB$	Finite-state

Context-free and regular grammars are used in practice

Notes

Are languages context-free?

Mostly true, with exceptions

Swiss German:
"...das mer d'chind em Hans es huus lönd häfte aastriiche"

Word-by-word:
"...that we the children Hans the house let help paint"

Translation:
"...that we let the children help Hans paint the house"

Notes

Parsers

- assign each word in a sentence to a *part of speech*
- originally developed for programming languages (no ambiguities)
- only available for context-free and regular grammars
- top-down: start with *S* and generate rules until you reach the words (terminal symbols)
- bottom-up: start with the words and work your way up until you reach *S*

Notes

Example: Top-down parser

Parts of speech	Rules
S	
NP VP	S → NP VP
NAME VP	NP → NAME
Mary VP	NAME → Mary
Mary V NP	VP → V NP
Mary loves NP	V → loves
Mary loves ADJ NP1	NP → ADJ NP1
Mary loves that NP1	ADJ → that
Mary loves that N	NP1 → N
Mary loves that person	N → person

Notes

Example: Bottom-up parser

Parts of speech	Rules
Mary loves that person	
NAME loves that person	NAME → Mary
NAME V that person	V → loves
NAME V ADJ person	ADJ → that
NAME V ADJ N	N → person
NP V ADJ N	NP → NAME
NP V ADJ NP1	NP1 → N
NP V NP	NP → ADJ NP1
NP VP	VP → V NP
S	S → NP VP

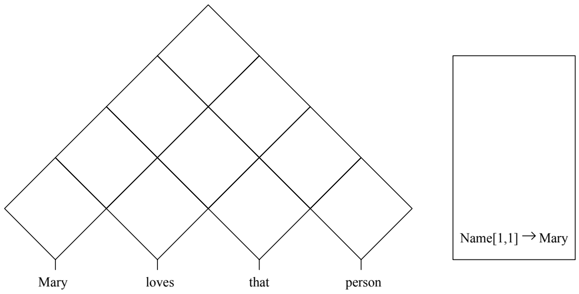
Notes

Top-down vs bottom-up parsers

- Top-down characteristics:
  - + very predictive
  - + only consider grammatical combinations
  - − predict constituents that do not have a match in the text
- Bottom-up characteristics:
  - + check input text only once
  - + suitable for robust language processing
  - − may build trees that do not lead to full parse
- All in all, similar performance

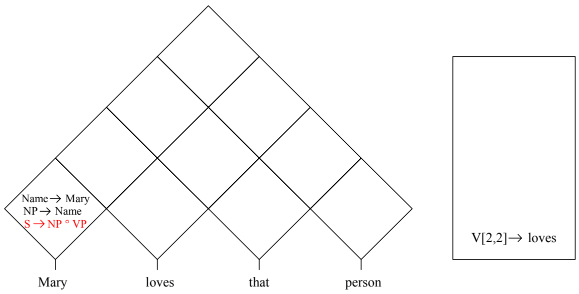
Notes

Chart parsing (dynamic programming)



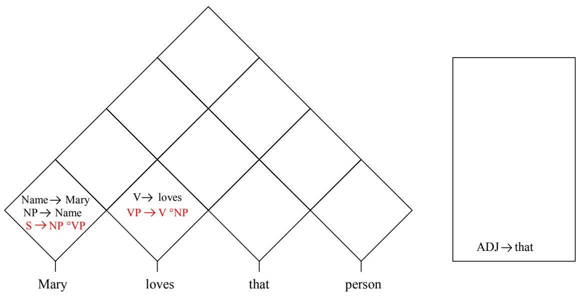
Notes

Chart parsing (dynamic programming)



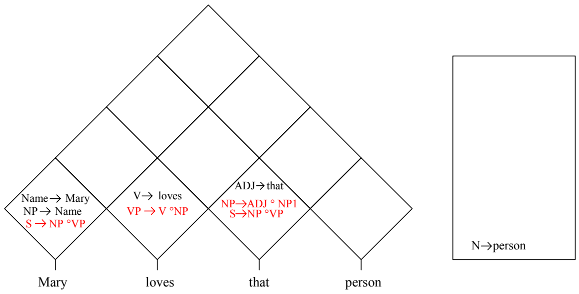
Notes

Chart parsing (dynamic programming)



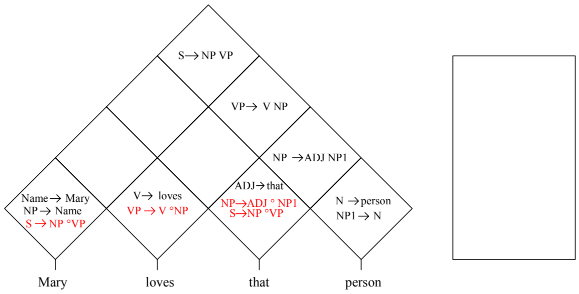
Notes

Chart parsing (dynamic programming)



Notes

Chart parsing (dynamic programming)



Notes

Stochastic Language Models (SLM)

1. formal grammars lack coverage (for general domains)
2. spoken language does not follow strictly the grammar

Model sequences of words statistically:

$$P(W) = P(w_1 w_2 \dots w_n)$$

Notes

Probabilistic Context-free grammars (PCFGs)

Assign probabilities to generative rules:

$$P(A \rightarrow \alpha | G)$$

Then calculate probability of generating a word sequence $w_1 w_2 \dots w_n$ as probability of the rules necessary to go from S to $w_1 w_2 \dots w_n$:

$$P(S \Rightarrow w_1 w_2 \dots w_n | G)$$

Notes

Training PCFGs

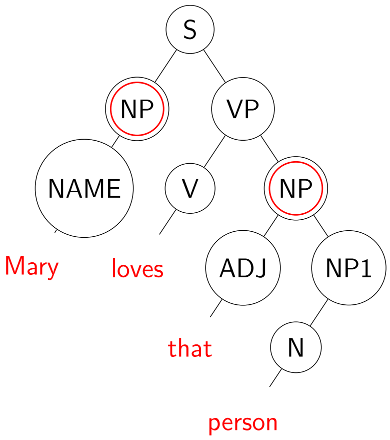
If annotated corpus, Maximum Likelihood estimate:

$$P(A \to \alpha_j) = \frac{C(A \to \alpha_j)}{\sum_{i=1}^{m} C(A \to \alpha_i)}$$

If non-annotated corpus: **inside-outside algorithm** (similar to HMM training, forward-backward)

Notes

Independence assumption

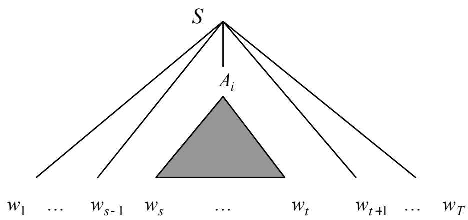


Notes

Inside-outside probabilities

Chomsky's normal forms: $A_i \to A_m A_n$ or $A_i \to w_l$

$$\begin{aligned} \text{inside}(s, A_i, t) &= P(A_i \Rightarrow w_s w_{s+1} \dots w_t) \\ \text{outside}(s, A_i, t) &= P(S \Rightarrow w_1 \dots w_{s-1}\ A_i\ w_{t+1} \dots w_T) \end{aligned}$$



Notes

Probabilistic Context-free grammars:limitations

- probabilities help sorting alternative explanations, but
- still problem with coverage: the production rules are hand made

$$P(A \to \alpha | G)$$

Notes

N-gram Language Models

Flat model: no hierarchical structure

$$\begin{aligned} P(\mathbf{W}) &= P(w_1, w_2, \ldots, w_n) \\ &= P(w_1)P(w_2|w_1)P(w_3|w_1, w_2)\cdots P(w_n|w_1, w_2\ldots, w_{n-1}) \\ &= \prod_{i=1}^{n} P(w_i|w_1, w_2, \ldots, w_{i-1}) \end{aligned}$$

Approximations:

$P(w_i\|w_1, w_2, \ldots, w_{i-1}) = P(w_i)$		(Unigram)
$P(w_i\|w_1, w_2, \ldots, w_{i-1}) = P(w_i\|w_{i-1})$		(Bigram)
$P(w_i\|w_1, w_2, \ldots, w_{i-1}) = P(w_i\|w_{i-2}, w_{i-1})$		(Trigram)
$P(w_i\|w_1, w_2, \ldots, w_{i-1}) = P(w_i\|w_{i-N+1}, \ldots, w_{i-1})$		(N-gram)

Notes

Example (Bigram)

$$\begin{gathered} P(Mary, loves, that, person) = \\ P(Mary|<\mathrm{s}>)P(loves|Mary)P(that|loves) \\ P(person|that)P(</\mathrm{s}>|person) \end{gathered}$$

Notes

N-gram estimation (Maximum Likelihood)

$$P(w_i|w_{i-N+1}, \ldots, w_{i-1}) = \frac{C(\overbrace{w_{i-N+1}, \ldots, w_{i-1}, w_i}^{N})}{C(\underbrace{w_{i-N+1}, \ldots, w_{i-1}}_{N-1})}$$

$$= \frac{C(w_{i-N+1}, \ldots, w_{i-1}, w_i)}{\sum_{w_i} C(w_{i-N+1}, \ldots, w_{i-1}, w_i)}$$

Problem: **data sparseness**

Notes

N-gram estimation example

Corpus:
1: John read her book
2: I read a different book
3: John read a book by Mulan

$$\begin{aligned} P(\text{John}|<\text{s}>) &= \frac{C(<\text{s}>,\text{John})}{C(<\text{s}>)} &= \frac{2}{3} \\ P(\text{read}|\text{John}) &= \frac{C(\text{John},\text{read})}{C(\text{John})} &= \frac{2}{2} \\ P(\text{a}|\text{read}) &= \frac{C(\text{read},\text{a})}{C(\text{read})} &= \frac{2}{3} \\ P(\text{book}|\text{a}) &= \frac{C(\text{a},\text{book})}{C(\text{a})} &= \frac{1}{2} \\ P(</\text{s}>|\text{book}) &= \frac{C(\text{book},</\text{s}>)}{C(\text{book})} &= \frac{2}{3} \end{aligned}$$

$$\begin{aligned} P(\text{John}, \text{read}, \text{a}, \text{book}) = {} & P(\text{John}|<\text{s}>)P(\text{read}|\text{John})P(\text{a}|\text{read})\cdots \\ & P(\text{book}|\text{a})P(</\text{s}>|\text{book}) = 0.148 \\ P(\text{Mulan}, \text{read}, \text{a}, \text{book}) = {} & P(\text{Mulan}|<\text{s}>)\cdots = 0 \end{aligned}$$

Notes

N-gram Smoothing

Problem:

- Many very possible word sequences may have been observed in zero or very low numbers in the training data
- Leads to extremely low probabilities, effectively disabling this word sequence, no matter how strong the acoustic evidence is

Solution: smoothing

- produce more robust probabilities for unseen data at the cost of modelling the training data slightly worse

Notes

Simplest Smoothing technique

Instead of ML estimate

$$P(w_i|w_{i-N+1},\ldots,w_{i-1}) = \frac{C(w_{i-N+1},\ldots,w_{i-1},w_i)}{\sum_{w_i} C(w_{i-N+1},\ldots,w_{i-1},w_i)}$$

Use

$$P(w_i|w_{i-N+1},\ldots,w_{i-1}) = \frac{1 + C(w_{i-N+1},\ldots,w_{i-1},w_i)}{\sum_{w_i} (1 + C(w_{i-N+1},\ldots,w_{i-1},w_i))}$$

- prevents zero probabilities
- but still very low probabilities

Notes

N-gram simple smoothing example

Corpus:
1: John read her book
2: I read a different book
3: John read a book by Mulan

$$P(\text{John}|<\text{s}>) = \frac{1+C(<\text{s}>,\text{John})}{11+C(<\text{s}>)} = \frac{3}{14}$$

$$P(\text{read}|\text{John}) = \frac{1+C(\text{John},\text{read})}{11+C(\text{John})} = \frac{3}{13}$$

$\ldots$

$$P(\text{Mulan}|<\text{s}>) = \frac{1+C(<\text{s}>,\text{Mulan})}{11+C(<\text{s}>)} = \frac{1}{14}$$

$$P(\text{John},\text{read},\text{a},\text{book}) = P(\text{John}|<\text{s}>)P(\text{read}|\text{John})P(\text{a}|\text{read})\cdots P(\text{book}|\text{a})P(</\text{s}>|\text{book}) = 0.00035(0.148)$$

$$P(\text{Mulan},\text{read},\text{a},\text{book}) = P(\text{Mulan}|<\text{s}>)P(\text{read}|\text{Mulan})P(\text{a}|\text{read})\cdots P(\text{book}|\text{a})P(</\text{s}>|\text{book}) = 0.000084(0)$$

Notes

Interpolation vs Backoff smoothing

Interpolation models:

- Linear combination with lower order n-grams
- Modifies the probabilities of both nonzero and zero count n-grams

Backoff models:

- Use lower order n-grams when the requested n-gram has zero or very low count in the training data
- Nonzero count n-grams are unchanged
- Discounting: Reduce the probability of seen n-grams and distribute among unseen ones

Notes

Interpolation vs Backoff smoothing

Interpolation models:

$$P_{\text{smooth}}(w_i|w_{i-N+1},\ldots,w_{i-1}) = \lambda \overbrace{P_{\text{ML}}(w_i|w_{i-N+1},\ldots,w_{i-1})}^{N} + (1-\lambda)\overbrace{P_{\text{smooth}}(w_i|w_{i-N+2},\ldots,w_{i-1})}^{N-1}$$

Backoff models:

$$P_{\text{smooth}}(w_i|w_{i-N+1},\ldots,w_{i-1}) = \begin{cases} \alpha \overbrace{P(w_i|w_{i-N+1},\ldots,w_{i-1})}^{N} & \text{if } C(w_i|w_{i-N+1},\ldots,w_{i-1}) > 0 \\ \gamma \overbrace{P_{\text{smooth}}(w_i|w_{i-N+2},\ldots,w_{i-1})}^{N-1} & \text{if } C(w_i|w_{i-N+1},\ldots,w_{i-1}) = 0 \end{cases}$$

Notes

Deleted interpolation smoothing

Recursively interpolate with n-grams of lower order:
if $\text{history}_n = w_{i-n+1},\ldots,w_{i-1}$

$$P_I(w_i|\text{history}_n) = \lambda_{\text{history}_n} P(w_i|\text{history}_n) + (1-\lambda_{\text{history}_n}) P_I(w_i|\text{history}_{n-1})$$

- hard to estimate $\lambda_{\text{history}_n}$ for every history
- cluster into moderate number of weights

Notes

Backoff smoothing

Use $P(w_i|\text{history}_{n-1})$ only if you lack data for $P(w_i|\text{history}_n)$

Notes

Good-Turing estimate

- Partition n-grams into groups depending on their frequency in the training data
- Change the number of occurrences of an n-gram according to

$$r^* = (r+1)\frac{n_{r+1}}{n_r}$$

where r is the occurrence number, n_r is the number of n-grams that occur r times

Notes

Katz smoothing

based on Good-Turing: combine higher and lower order n-grams

For every N-gram:

1. if count r is large (> 5 or 8), do not change it
2. if count r is small but non-zero, discount with $\approx r^*$
3. if count $r = 0$, reassign discounted counts with lower order N-gram

$$C^*(w_{i-1}, w_i) = \alpha(w_{i-1})P(w_i)$$

Notes

Kneser-Ney smoothing: motivation

Background

- Lower order n-grams are often used as backoff model if the count of a higher-order n-gram is too low (e.g. unigram instead of bigram)

Problem

- Some words with relatively high unigram probability only occur in a few bigrams. E.g. Francisco, which is mainly found in San Francisco. However, infrequent word pairs, such as New Francisco, will be given too high probability if the unigram probabilities of New and Francisco are used. Maybe instead, the Francisco unigram should have a lower value to prevent it from occurring in other contexts.

I can't see without my reading...

Notes

Kneser-Ney intuition

If a word has been seen in many contexts it is more likely to be seen in new contexts as well.

- instead of backing off to lower order n-gram, use continuation probability

Example: instead of unigram $P(w_i)$, use

$$P_{CONTINUATION}(w_i) = \frac{|\{w_{i-1} : C(w_{i-1}w_i) > 0\}|}{\sum_{w_i} |\{w_{i-1} : C(w_{i-1}w_i) > 0\}|}$$

I can't see without my reading... glasses

Notes

Class N-grams

1. Group words into semantic or grammatical classes
2. build n-grams for class sequences:

$$P(w_i|c_{i-N+1} \dots c_{i-1}) = P(w_i|c_i)P(c_i|c_{i-N+1} \dots c_{i-1})$$

- rapid adaptation, small training sets, small models
- works on limited domains
- classes can be rule-based or data-driven

Notes

Combining PCFGs and N-grams

Only N-grams:

Meeting at three with Zhou Li
Meeting at four PM with Derek

$P(\text{Zhou}|\text{three}, \text{with})$ and $P(\text{Derek}|\text{PM}, \text{with}))$

N-grams + CFGs:

Meeting {at three: TIME} with {Zhou Li: NAME}
Meeting {at four PM: TIME} with {Derek: NAME}

$P(\text{NAME}|\text{TIME}, \text{with})$

Notes

Adaptive Language Models

- conversational topic is not stationary
- topic stationary over some period of time
- build more specialised models that can adapt in time

Techniques

- Cache Language Models
- Topic-Adaptive Models
- Maximum Entropy Models

Notes

Cache Language Models

1. build a full static n-gram model
2. during conversation accumulate low order n-grams
3. interpolate between 1 and 2

Notes

Topic-Adaptive Models

1. cluster documents into topics (manually or data-driven)
2. use information retrieval techniques with current recognition output to select the right cluster
3. if off-line run recognition in several passes

Notes

Maximum Entropy Models

Instead of linear combination:

1. reformulate information sources into constraints
2. choose maximum entropy distribution that satisfies the constraints

Constraints general form:

$$\sum_X P(X) f_i(X) = E_i$$

Example: unigram

$$f_{w_i} = \begin{cases} 1 & \text{if } w = w_i \\ 0 & \text{otherwise} \end{cases}$$

Notes

Language Model Evaluation

- Evaluation in combination with Speech Recogniser
  - hard to separate contribution of the two
- Evaluation based on probabilities assigned to text in the training and test set

Notes

Information, Entropy, Perplexity

Information:

$$I(x_i) = \log \frac{1}{P(x_i)}$$

Entropy:

$$H(X) = E[I(X)] = -\sum_i P(x_i) \log P(x_i)$$

Perplexity:

$$PP(X) = 2^{H(X)}$$

Notes

Perplexity of a model

We do not know the “true” distribution $p(w_1, \dots, w_n)$. But we have a model $m(w_1, \dots, w_n)$. The cross-entropy is:

$$H(p, m) = -\sum_{w_1,\dots,w_n} p(w_1, \dots, w_n) \log m(w_1, \dots, w_n)$$

Cross-entropy is upper bound to entropy:

$$H \leq H(p, m)$$

The better the model, the lower the cross-entropy and the lower the perplexity (on the same data)

Notes

Test-set Perplexity

Notes

Estimate the distribution $p(w_1, \ldots, w_n)$ on the training data
Evaluate it on the test data

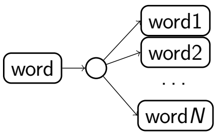
$$H = -\sum_{w_1,\ldots,w_n \in \text{test set}} p(w_1, \ldots, w_n) \log p(w_1, \ldots, w_n)$$

$$PP = 2^H$$

Perplexity and branching factor

Notes

Perplexity is roughly the geometric mean of the branching factor



Shannon: 2.39 for English letters and 130 for English words
Digit strings: 10
n-gram English: 50–1000
Wall Street Journal test set: 180 (bigram) 91 (trigram)

Performance of N-grams

Notes

Models	Perplexity	Word Error Rate
Unigram Katz	1196.45	14.85%
Unigram Kneser-Ney	1199.59	14.86%
Bigram Katz	176.31	11.38%
Bigram Kneser-Ney	176.11	11.34%
Trigram Katz	95.19	9.69%
Trigram Kneser-Ney	91.47	9.60%

Wall Street Journal database Dictionary: 60 000 words
Training set: 260 000 000 words

Notes