Statistical Language Models

Lluís Padró
padro@lsi.upc.edu
TALP Research Center
Universitat Politècnica de Catalunya

Statistical NLP

Broad multidisciplinary area

- Linguistics to provide models of language
- Psychology to provide models of cognitive processes
- Information theory to provide models of communication
- Mathematics & Statistics to provide tools to analyze and acquire such models
- Computer Science to implement computable models

Problems of the traditional approach (1)

- Language Acquisition:
  Children try and discard syntax rules progressively
- Language Change:
  Language changes along time (*ale* vs. *eel*, *while* as Adv vs. Noun, *near* as Prep vs. Adj)
- Language Variation:
  Dialect continuum (e.g. Inuit)
- Language is a collection of statistical distributions:
  Weights for rules (phonetic, syntactic, etc) change when learning, along time, between communities...

Problems of the traditional approach (2)

- Structural ambiguity

  Our company is training workers — *Parker saw Mary*

  Our problem is training workers — *The a are of I*

  Our product is training wheels
- Scalability: scaling up from small and domain specific applications
- Practicallity: Time costly to build systems with good coverage
- Brittleness: understanding metaphors
- Reasoning: Requires world knowledge and common sense knowledge $\Rightarrow$ learning

How Statistics helps

- Disambiguation: Stochastic grammars. *John walks*
- Degrees of grammaticality
- Naturalness: *strong tea*, *powerful car*
- Structural preferences:
  The emergency crews hate most is domestic violence
- Error tolerance:
  We sleeps *Thanks for all you help*
- Learning on the fly:
  One hectare is a hundred ares
  The are a of I
- Lexical Acquisition.

Zipf's Laws (1929)

- Word frequency is inversely proportional to its rank (speaker/hearer minimum effort) $f \sim 1/r$
- Number of senses is proportional to frequency root $m \sim \sqrt{f}$
- Frequency of intervals between repetitions is inversely proportional to the length of the interval $F \sim 1/I$
- Random generated languages satisfy Zipf's laws
- Frequency based approaches are hard, since most words are rare
    - Most common 5% words account for about 50% of a text
    - 90% least common words account for less than 10% of the text
    - Almost half of the words in a text occurr only once

Usual Objections

Stochastic models are for engineers, not for scientists

- Approximation to handle information impractical to collect in cases where initial conditions cannot be exactly determined (e.g. as queue theory models dynamical systems).
- If the system is not deterministic (i.e. has *emergent* properties), an stochastic account is more insightful than a reductionistic approach (e.g. statistical mechanics)

Chomsky's heritage: Statistics can not capture NL structure

- Techniques to estimate probabilities of unseen events.
- Chomsky's criticisms can be applied to Finite State, N-gram or Markov models, but not to all stochastic models.

Conclusions

- Statistical methods are relevant to language acquisition, change, variation, generation and comprehension.
- Pure algebraic methods are inadequate for understanding many important properties of language, such as the measure of goodness that allows to identify the correct parse among a large candidate set.
- The focus of computational linguistics has been up to now on technology, but the same techniques promise progress at unanswered questions about the nature of language.

Basics

Introduction
Basics
Statistical Models for NLP
Maximum Likelihood Estimation (MLE)
Maximum Entropy Modeling
Markovian Models
Similarity and Clustering
References

- Random variable: Function on a stochastic process.
  $X : \Omega \longrightarrow \mathcal{R}$
- Continuous and discrete random variables.
- Probability mass (or density) function, Frequency function:
  $p(x) = P(X = x)$.
  Discrete R.V.: $\sum_x p(x) = 1$
  Continuous R.V: $\int_{-\infty}^{\infty} p(x)dx = 1$
- Distribution function: $F(x) = P(X \leq x)$
- Expectation and variance, standard deviation
  $E(X) = \mu = \sum_x xp(x)$
  $VAR(X) = \sigma^2 = E((X - E(X))^2) = \sum_x (x - \mu)^2 p(x)$

Joint and Conditional Distributions

- Joint probability mass function: $p(x, y)$
- Marginal distribution:

$$p_X(x) = \sum_y p(x, y) \qquad p_{X|Y}(x \mid y) = \frac{p(x, y)}{p_Y(y)}$$
$$p_Y(y) = \sum_x p(x, y)$$

Simplified Polynesian. Sequences of C-V syllabes: Two random variables C,V

P(C,V)	p	t	k	
a	1/16	3/8	1/16	1/2
i	1/16	3/16	0	1/4
u	0	3/16	1/16	1/4
	1/8	3/4	1/8	

$P(p \mid i) = ?$
$P(a \mid t \vee k) = ?$
$P(a \vee i \mid p) = ?$

Samples and Estimators

- Random samples
- Sample variables:

  Sample mean: $\bar{\mu}_n = \frac{1}{n}\sum_{i=1}^{n} x_i$

  Sample variance: $s_n^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{\mu}_n)^2.$
- Law of Large Numbers: as n increases, $\bar{\mu}_n$ and s_n^2 converge to μ and σ^2
- Estimators: Sample variables used to estimate real parameters.

Finding good estimators: MLE

Maximum Likelihood Estimation (MLE)

- Choose the alternative that maximizes the probability of the observed outcome.
- $\bar{\mu}_n$ is a MLE for $E(X)$
- s_n^2 is a MLE for σ^2
- Data sparseness problem. Smoothing tecnhiques.

$P(a,b)$	dans	en	à	sur	au-cours-de	pendant	selon	
in	0.04	0.10	0.15	0	0.08	0.03	0	0.40
on	0.06	0.25	0.10	0.15	0	0	0.04	0.60
total	0.10	0.35	0.25	0.15	0.08	0.03	0.04	1.0

Finding good estimators: MEE

Maximum Entropy Estimation (MEE)

- Choose the alternative that maximizes the entropy of the obtained distribution, maintaning the observed probabilities.

Observations:
$p(en \vee à) = 0.6$

$P(a,b)$	dans	en	à	sur	au-cours-de	pendant	selon	
in	0.04	0.15	0.15	0.04	0.04	0.04	0.04	
on	0.04	0.15	0.15	0.04	0.04	0.04	0.04	
total		0.6						1.0

Finding good estimators: MEE

Maximum Entropy Estimation (MEE)

- Choose the alternative that maximizes the entropy of the obtained distribution, maintaning the observed probabilities.

Observations:
$p(en \vee \grave{a}) = 0.6; \qquad p((en \vee \grave{a}) \wedge in) = 0.4$

$P(a,b)$	dans	en	à	sur	au-cours-de	pendant	selon	
in	0.04	**0.20**	**0.20**	0.04	0.04	0.04	0.04	
on	0.04	0.10	0.10	0.04	0.04	0.04	0.04	
total		$\underbrace{\qquad\qquad}_{0.6}$						1.0

Finding good estimators: MEE

Maximum Entropy Estimation (MEE)

- Choose the alternative that maximizes the entropy of the obtained distribution, maintaning the observed probabilities.

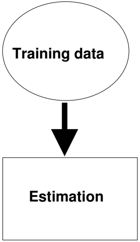
Observations:

$p(en \vee \grave{a}) = 0.6; \qquad p((en \vee \grave{a}) \wedge in) = 0.4; \qquad p(in) = 0.5$

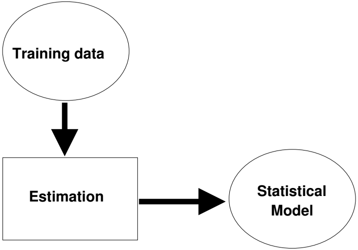
$P(a,b)$	dans	en	à	sur	au-cours-de	pendant	selon	
in	0.02	**0.20**	**0.20**	0.02	0.02	0.02	0.02	**0.5**
on	0.06	0.10	0.10	0.06	0.06	0.06	0.06	
total		$\underbrace{\qquad\qquad}_{\mathbf{0.6}}$						1.0

Statistical models for NLP

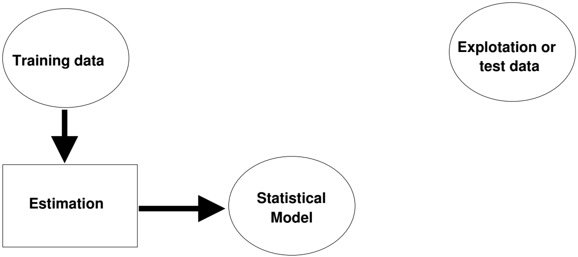
Statistical models for NLP



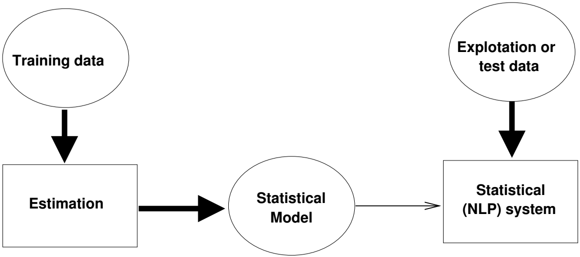
Statistical models for NLP



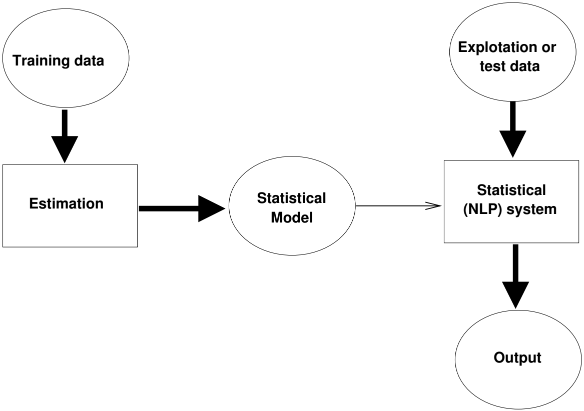
Statistical models for NLP



Statistical models for NLP



Statistical models for NLP



Prediction Models & Similarity Models

- Prediction Models: Able to *predict* probabilities of future events, knowing past and present.
- Similarity Models: Able to compute *similarities* between objects (may be used to predict, EBL).

Similarity Models

- Objects represented as feature-vectors, feature-sets, distribution-vectors.
- Used to group objects (clustering, data analysis, pattern discovery, ...)
- If existig objects are classified, similarity may be used as a prediction (example-based ML techniques).
- Example: Document representation
  - Documents are represented as vectors in a high dimensional $\mathbb{R}^n$ space.
  - Dimensions are word forms, lemmas, NEs, n-grams, ...
  - Values may be either binary or real–valued (count, frequency, ...)
  - Vector space algebra and metrics can be used

$$\vec{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix} \qquad \vec{x}^T = [x_1 \ldots x_N] \qquad |\vec{x}| = \sqrt{\sum_{i=1}^{N} x_i^2}$$

Prediction Models

Example: Noisy Channel Model (Shannon 48)

Input P(i) $\longrightarrow$ **Channel** P(o|i) $\longrightarrow$ **Output**

NLP Applications

Appl.	Input	Output	$p(i)$	$p(o \mid i)$
MT	L word sequence	M word sequence	$p(L)$	Translation model
OCR	Actual text	Text with mistakes	prob. of language text	model of OCR errors
PoS tagging	PoS tags sequence	word sequence	prob. of PoS sequence	$p(w \mid t)$
Speech recog.	word sequence	speech signal	prob. of word sequence	acoustic model

Given **o**, we want to find the most likely **i**

$$\operatorname*{argmax}_{\mathbf{i}} \Pr(\mathbf{i} \mid \mathbf{o}) = \operatorname*{argmax}_{\mathbf{i}} \Pr(\mathbf{o}, \mathbf{i}) = \operatorname*{argmax}_{\mathbf{i}} \Pr(\mathbf{i}) \Pr(\mathbf{o} \mid \mathbf{i})$$

Inference & Modeling

- Using data to infer information about distributions
  - Parametric / non-parametric estimation
  - Finding good estimators: MLE, MEE, ...
- Example: Language Modeling (Shannon game), N-gram models.
- Predictions based on past behaviour
  - Target / classification features → Independence assumptions
  - Equivalence classes (bins).
    Granularity: discrimination *vs.* statistical reliability

N-gram models

- Predicting the next word in a sequence, given the *history* or *context*. $P(w_n \mid w_1 \ldots w_{n-1})$
- Markov assumption: Only *local* context (of size $n-1$) is taken into account. $P(w_i \mid w_{i-n+1} \ldots w_{i-1})$
- bigrams, trigrams, four-grams ($n = 2, 3, 4$).
  Sue swallowed the large green $<$*?*$>$
- Parameter estimation (number of equivalence classes)
- Parameter reduction: stemming, semantic classes, PoS, …

Model	Parameters
bigram	$20,000^2 = 4 \times 10^8$
trigram	$20,000^3 = 8 \times 10^{12}$
four-gram	$20,000^4 = 1.6 \times 10^{17}$

Language model sizes for a 20,000 words vocabulary

MLE Overview

Estimate the probability of the target feature based on observed data. The prediction task can be reduced to having good estimations of the *n*-gram distribution:

$$P(w_n \mid w_1 \dots w_{n-1}) = \frac{P(w_1 \dots w_n)}{P(w_1 \dots w_{n-1})}$$

- **MLE (Maximum Likelihood Estimation)**

  $P_{MLE}(w_1 \dots w_n) = \frac{C(w_1 \dots w_n)}{N}$

  $P_{MLE}(w_n \mid w_1 \dots w_{n-1}) = \frac{C(w_1 \dots w_n)}{C(w_1 \dots w_{n-1})}$

  - No probability mass for unseen events
  - Unsuitable for NLP
  - Data sparseness, Zipf's Law

Notation

- $C(w_1 \dots w_n)$: Observed occurrence count for n-gram $w_1 \dots w_n$.
- $C_A(w_1 \dots w_n)$: Observed occurrence count for n-gram $w_1 \dots w_n$ on data subset A.
- N: Number of observed n-gram occurrences

$$N = \sum_{w_1 \dots w_n} C(w_1 \dots w_n)$$

- N_k: Number of classes (n-grams) observed k times.
- N_k^A: Number of classes (n-grams) observed k times on data subset A.
- B: Number of equivalence classes or bins (number of potentially observable n-grams).

Smoothing 1 - Adding Counts

- **Laplace's Law** (adding one)

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

  - For large values of B too much probability mass is assigned to unseen events

- **Lidstone's Law**

$$P_{LID}(w_1 \ldots w_n) = \frac{C(w_1 \ldots w_n) + \lambda}{N + B\lambda}$$

  - Usually $\lambda = 0.5$, *Expected Likelihood Estimation*.
  - Equivalent to linear interpolation between MLE and uniform prior, with $\mu = N/(N + B\lambda)$,

$$P_{LID}(w_1 \ldots w_n) = \mu \frac{C(w_1 \ldots w_n)}{N} + (1 - \mu)\frac{1}{B}$$

Smoothing 2 - Discounting Counts

- **Absolute Discounting**

$$P_{ABS}(w_1 \ldots w_n) = \begin{cases} \frac{r-\delta}{N} & \text{if } r > 0 \\ \frac{(B-N_0)\delta/N_0}{N} & \text{otherwise} \end{cases}$$

- **Linear Discounting**

$$P_{LIN}(w_1 \ldots w_n) = \begin{cases} \frac{(1-\alpha)r}{N} & \text{if } r > 0 \\ \frac{\alpha}{N_0} & \text{otherwise} \end{cases}$$

Smoothing 3 - Held Out Data

- *Notation:* γ stands for $w_1 \ldots w_n$.
- Divide the train corpus in two subsets, A and B.
- Define: $T_r^{AB} = \sum_{\gamma : C_A(\gamma)=r} C_B(\gamma)$
- **Held Out Estimator**

$$P_{HO}(w_1 \ldots w_n) = \frac{T^{AB}_{C_A(\gamma)}}{N^A_{C_A(\gamma)}} \times \frac{1}{N}$$

- **Cross Validation** (deleted estimation)

$$P_{DEL}(w_1 \ldots w_n) = \frac{T^{AB}_{C_A(\gamma)} + T^{BA}_{C_B(\gamma)}}{N^A_{C_A(\gamma)} + N^B_{C_B(\gamma)}} \times \frac{1}{N}$$

- **Cross Validation** (Leave-one-out)

Combining Estimators

- **Simple Linear Interpolation**
  $$P_{LI}(w_n \mid w_{n-2}, w_{n-1}) = \\ = \lambda_1 P_1(w_n) + \lambda_2 P_2(w_n \mid w_{n-1}) + \lambda_3 P_3(w_n \mid w_{n-2}, w_{n-1})$$
- **General Linear Interpolation**
  $$P_{LI}(w_n \mid h) = \sum_{i=1}^{k} \lambda_i(h) P_i(w \mid h_i)$$
- **Katz's Backing-off**
  $$P_{BO}(w_i \mid w_{i-n+1} \dots w_{i-1}) = \begin{cases} (1 - d_{w_{i-n+1} \dots w_{i-1}}) \dfrac{C(w_{i-n+1} \dots w_i)}{C(w_{i-n+1} \dots w_{i-1})} & \textit{if } C(w_{i-n+1} \dots w_i) > k \\ \alpha_{w_{i-n+1} \dots w_{i-1}} P_{BO}(w_i \mid w_{i-n+2} \dots w_{i-1}) & \textit{otherwise} \end{cases}$$

MEM Overview

- Maximum Entropy: alternative estimation technique.
- Able to deal with different kinds of evidence
- ME principle:
  - Do not assume anything about non-observed events.
  - Find the most uniform (maximum entropy, less informed) probability distribution that matches the observations.
- Example:

$p(a,b)$	0	1	
x	?	?	
y	?	?	
total	0.6		1.0

Observations

$p(a,b)$	0	1	
x	0.5	0.1	
y	0.1	0.3	
total	0.6		1.0

One possible $p(a,b)$

$p(a,b)$	0	1	
x	0.3	0.2	
y	0.3	0.2	
total	0.6		1.0

Max.Entropy $p(a,b)$

ME Modeling

- Observed facts are constraints for the desired model p.
- Constraints take the form of feature functions:

$$f_i : \varepsilon \to \{0, 1\}$$

- The desired model must satisfy the constraints:

$$E_p(f_i) = E_{\widetilde{p}}(f_i) \quad \forall i$$

where:

$$E_p(f_i) = \sum_{x \in \varepsilon} p(x) f_i(x)$$ expectation of model p.

$$E_{\widetilde{p}}(f_i) = \sum_{x \in \varepsilon} \widetilde{p}(x) f_i(x)$$ observed expectation.

Example

- Example:

$$\varepsilon = \{x, y\} \times \{0, 1\}$$

$p(a,b)$	0	1	
x	?	?	
y	?	?	
total	0.6		1.0

- Observed fact: $p(x,0) + p(y,0) = 0.6$
- Encoded as a constraint: $E_p(f_1) = 0.6$
  where:
  - $f_1(a,b) = \begin{cases} 1 & \textit{if } b = 0 \\ 0 & \textit{otherwise} \end{cases}$
  - $E_p(f_1) = \sum_{(a,b) \in \{x,y\} \times \{0,1\}} p(a,b) f_1(a,b)$

Probability Model

- There is an infinite set P of probability models consistent with observations:

$$P = \{p \mid E_p(f_i) = E_{\widetilde{p}}(f_i),\ \forall i = 1 \ldots k\}$$

- Maximum entropy model

$$p^* = \operatorname*{argmax}_{p \in P} H(p)$$

$$H(p) = -\sum_{x \in \varepsilon} p(x) \log p(x)$$

Conditional Probability Model

- For NLP applications, we are usually interested in conditional distributions $P(A|B)$, thus:

$$E_{\widetilde{p}}(f_j) = \sum_{a,b} \widetilde{p}(a,b) f_j(a,b)$$

$$E_p(f_j) = \sum_{a,b} \widetilde{p}(b) p(a \mid b) f_j(a,b)$$

- Maximum entropy model

$$p^* = \operatorname*{argmax}_{p \in P} H(p)$$

$$H(p) = H(A \mid B) = -\sum_{a,b} \widetilde{p}(b) p(a \mid b) \log p(a \mid b)$$

Parameter Estimation

Example: Maximum entropy model for translating *in* to French

- No constraints

$P(x)$	dans	en	à	au-cours-de	pendant	
	0.2	0.2	0.2	0.2	0.2	
total						1.0

- With constraint $p(dans) + p(en) = 0.3$

$P(x)$	dans	en	à	au-cours-de	pendant	
	0.15	0.15	0.233	0.233	0.233	
total	**0.3**					1.0

- With constraints $p(dans) + p(en) = 0.3;\ p(en) + p(à) = 0.5$

...Not so easy !

Parameter estimation

- Exponential models. (Lagrange multipliers optimization)
  $p(a \mid b) = \frac{1}{Z(b)} \prod_{j=1}^{k} \alpha_j^{f_j(a,b)} \qquad \alpha_j > 0$
  $Z(b) = \sum_a \prod_{i=1}^{k} \alpha_i^{f_i(a,b)}$
- also formuled as
  $p(a \mid b) = \frac{1}{Z(b)} \exp(\sum_{j=1}^{k} \lambda_j f_j(a,b))$
  $\lambda_i = \ln \alpha_i$
- Each model parameter weights the influence of a feature.
- Optimal parameters (ME model) can be computed with:
  - GIS. Generalized Iterative Scaling(Darroch & Ratcliff 72)
  - IIS. Improved Iterative Scaling (Della Pietra et al. 96)
  - LM-BFGS. Limited Memory BFGS (Malouf 03)

Improved Iterative Scaling (IIS)

Input: Feature functions $f_1 \dots f_n$, empirical distribution $\widetilde{p}(a, b)$
Output: λ_i^* parameters for optimal model p^*

Start with $\lambda_i = 0$ for all $i \in \{1 \dots n\}$
Repeat
  For each $i \in \{1 \dots n\}$ **do**
    let $\Delta\lambda_i$ be the solution to

$$\sum_{a,b} \widetilde{p}(b) p(a \mid b) f_i(a, b) \exp(\Delta\lambda_i \sum_{j=1}^{n} f_j(a, b)) = \widetilde{p}(f_i)$$

    $\lambda_i \leftarrow \lambda_i + \Delta\lambda_i$
  end for
Until all λ_i have converged

Application to NLP Tasks

- Speech processing (Rosenfeld 94)
- Machine Translation (Brown et al 90)
- Morphology (Della Pietra et al. 95)
- Clause boundary detection (Reynar & Ratnaparkhi 97)
- PP-attachment (Ratnaparkhi et al 94)
- PoS Tagging (Ratnaparkhi 96, Black et al 99)
- Partial Parsing (Skut & Brants 98)
- Full Parsing (Ratnaparkhi 97, Ratnaparkhi 99)
- Text Categorization (Nigam et al 99)

PoS Tagging (Ratnaparkhi 96)

- Probabilistic model over $H \times T$

$$h_i = (w_i, w_{i+1}, w_{i+2}, w_{i-1}, w_{i-2}, t_{i-1}, t_{i-2})$$

$$f_j(h_i, t) = \begin{cases} 1 & \textit{if suffix}(w_i) = \texttt{ing} \wedge t = \texttt{VBG} \\ 0 & \textit{otherwise} \end{cases}$$

- Compute $p^*(h, t)$ using GIS
- Disambiguation algorithm: *beam search*

$$p(t \mid h) = \frac{p(h, t)}{\sum_{t' \in T} p(h, t')}$$

$$p(t_1 \dots t_n \mid w_1 \dots w_n) = \prod_{i=1}^{n} p(t_i \mid h_i)$$

Text Categorization (Nigam et al 99)

- Probabilistic model over $W \times C$

$$d = (w_1, w_2 \ldots w_N)$$

$$f_{w,c'}(d,c) = \begin{cases} \frac{N(d,w)}{N(d)} & \text{if } c = c' \\ 0 & \text{otherwise} \end{cases}$$

- Compute $p^*(c \mid d)$ using IIS
- Disambiguation algorithm: Select class with highest

$$P(c \mid d) = \frac{1}{Z(d)} exp(\sum_i \lambda_i f_i(d,c))$$

MEM Summary

- Advantages
  - Teoretically well founded
  - Enables combination of random context features
  - Better probabilistic models than MLE (no smoothing needed)
  - General approach (features, events and classes)
- Disadvantages
  - Implicit probabilistic model (joint or conditional probability distribution obtained from model parameters).
  - High computational cost of GIS and IIS.
  - Overfitting in some cases.

Graphical Models

- **Generative models:**
  - Bayes rule $\Rightarrow$ independence assumptions.
  - Able to *generate* data.
- **Conditional models:**
  - No independence assumptions.
  - Unable to generate data.

Most algorithms of both kinds make assumptions about the nature of the data-generating process, predefining a fixed model structure and only acquiring from data the distributional information.

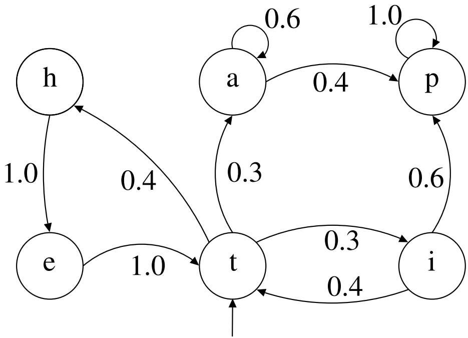
Usual Statistical Models in NLP

- **Generative models:**
  - Graphical: HMM (Rabiner 1990), IOHMM (Bengio 1996). Automata-learning algorithms: *No assumptions about model structure*. VLMM (Rissanen 1983), Suffix Trees (Galil & Giancarlo 1988), CSSR (Shalizi & Shalizi 2004).
  - Non-graphical: Stochastic Grammars (Lary & Young 1990)
- **Conditional models:**
  - Graphical: discriminative MM (Bottou 1991), MEMM (McCallum et al. 2000), CRF (Lafferty et al. 2001).
  - Non-graphical: Maximum Entropy Models (Berger et al 1996).

[Visible] Markov Models

- $X = (X_1, \ldots, X_T)$ sequence of random variables taking values in $S = \{s_1, \ldots, s_N\}$
- Markov Properties
  - Limited Horizon:
    $P(X_{t+1} = s_k \mid X_1, \ldots, X_t) = P(X_{t+1} = s_k \mid X_t)$
  - Time Invariant (Stationary):
    $P(X_{t+1} = s_k \mid X_t) = P(X_2 = s_k \mid X_1)$
- Transition matrix:
  $a_{ij} = P(X_{t+1} = s_j \mid X_t = s_i); \quad a_{ij} \geq 0, \ \forall i, j; \ \sum_{j=1}^{N} a_{ij} = 1, \ \forall i$
- Initial probabilities (or extra state s_0):
  $\pi_i = P(X_1 = s_i); \quad \sum_{i=1}^{N} \pi_i = 1$

MM Example



Sequence probability:

$$
\begin{aligned}
P(X_1,..,X_T) = &\\
&= P(X_1)P(X_2 \mid X_1)P(X_3 \mid X_1X_2)\ldots P(X_T \mid X_1..X_{T-1})\\
&= P(X_1)P(X_2 \mid X_1)P(X_3 \mid X_2)\ldots P(X_T \mid X_{T-1})\\
&= \pi_{X_1} \prod_{t=1}^{T-1} a_{X_tX_{t+1}}
\end{aligned}
$$

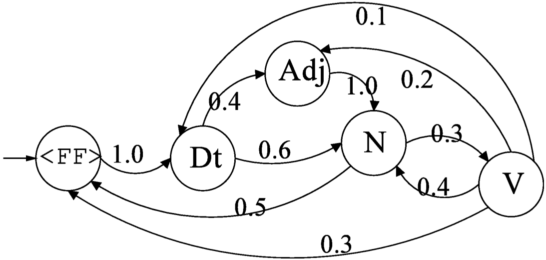
Hidden Markov Models (HMM)

- States and Observations
- Emission Probability:

$$b_{ik} = P(O_t = k \mid X_t = s_i)$$

- Used when underlying events probabilistically generate surface events:
  - PoS tagging (hidden states: PoS tags, observations: words)
  - ASR (hidden states: phonemes, observations: sound)
  - ...
- Trainable with unannotated data. Expectation Maximization (EM) algorithm.
- arc-emission *vs* state-emission

Example: PoS Tagging



Emission probabilities	.	the	this	cat	kid	eats	runs	fish	fresh	little	big
<FF>	1.0										
Dt		0.6	0.4								
N				0.6	0.1			0.3			
V						0.7	0.3				
Adj									0.3	0.3	0.4

HMM Fundamental Questions

Q1. **Observation probability (decoding):** Given a model $\mu = (A, B, \pi)$, how we do efficiently compute how likely is a certain observation ? That is, $P_\mu(O)$

Q2. **Classification:** Given an observed sequence O and a model μ, how do we choose the state sequence $(X_1, \ldots, X_T)$ that best explains the observations?

Q3. **Parameter estimation:** Given an observed sequence O and a space of possible models, each with different parameters (A, B, π), how do we find the model that best explains the observed data?

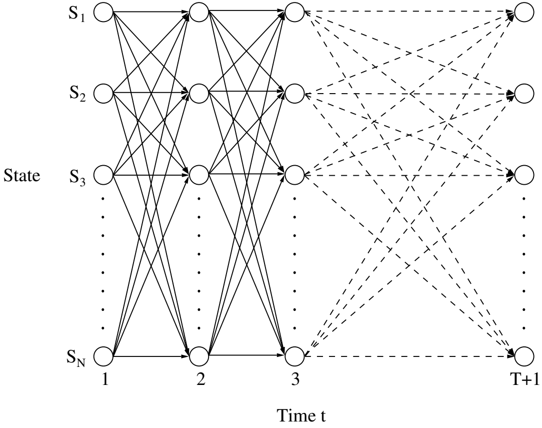
Question 1. Observation probability

- Let $O = (o_1, \ldots, o_T)$ observation sequence.
- For any state sequence $X = (X_1, \ldots, X_T)$, we have:

$$\begin{aligned} P_\mu(O \mid X) &= \prod_{t=1}^{T} P_\mu(o_t \mid X_t) \\ &= b_{X_1 o_1} b_{X_2 o_2} \ldots b_{X_T o_T} \end{aligned}$$

- $P_\mu(X) = \pi_{X_1} a_{X_1 X_2} a_{X_2 X_3} \ldots a_{X_{T-1} X_T}$
- $$\begin{aligned} P_\mu(O) &= \sum_X P_\mu(O, X) = \sum_X P_\mu(O \mid X) P_\mu(X) \\ &= \sum_{X_1 \ldots X_T} \pi_{X_1} b_{X_1 o_1} \prod_{t=2}^{T} a_{X_{t-1} X_t} b_{X_t o_t} \end{aligned}$$
- Complexity: $\mathcal{O}(TN^T)$
- Dynammic Programming: Trellis/lattice. $\mathcal{O}(TN^2)$

Trellis



Fully connected HMM where one can move from any state to any other at each step. A node $\{s_i, t\}$ of the trellis stores information about state sequences which include $X_t = i$.

Forward & Backward computation

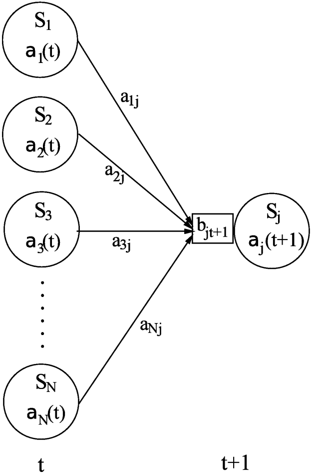
Forward procedure $\mathcal{O}(TN^2)$

We store $\alpha_i(t)$ at each trellis node $\{s_i, t\}$.

$\alpha_i(t) = P_\mu(o_1 \dots o_t, X_t = i)$ Probability of emmiting $o_1 \dots o_t$ and reach state s_i at time t.

1. Inicialization: $\alpha_i(1) = \pi_i b_{io_1}; \quad \forall i = 1 \dots N$
2. Induction: $\forall t : 1 \leq t < T$
$$\alpha_j(t+1) = \sum_{i=1}^{N} \alpha_i(t) a_{ij} b_{jo_{t+1}}; \quad \forall j = 1 \dots N$$
3. Total: $P_\mu(O) = \sum_{i=1}^{N} \alpha_i(T)$

Forward computation



Closeup of the computation of forward probabilities at one node. The forward probability $\alpha_j(t+1)$ is calculated by summing the product of the probabilities on each incoming arc with the forward probability of the originating node.

Forward & Backward computation

Backward procedure $\mathcal{O}(TN^2)$

We store $\beta_i(t)$ at each trellis node $\{s_i, t\}$.

$$\beta_i(t) = P_\mu(o_{t+1} \dots o_T \mid X_t = i)$$

Probability of emmiting $o_{t+1} \dots o_T$ given we are in state s_i at time t.

1. Inicialization: $\beta_i(T) = 1 \quad \forall i = 1 \dots N$
2. Induction: $\forall t : 1 \leq t < T$
$$\beta_i(t) = \sum_{j=1}^{N} a_{ij} b_{jo_{t+1}} \beta_j(t+1) \qquad \forall i = 1 \dots N$$
3. Total: $P_\mu(O) = \sum_{i=1}^{N} \pi_i b_{io_1} \beta_i(1)$

Forward & Backward computation

Combination

$$\begin{aligned} P_\mu(O, X_t = i) &= P_\mu(o_1 \dots o_{t-1}, X_t = i, o_t \dots o_T) \\ &= \alpha_i(t)\beta_i(t) \end{aligned}$$

$$P_\mu(O) = \sum_{i=1}^{N} \alpha_i(t)\beta_i(t) \qquad \forall t : 1 \leq t \leq T$$

Forward and Backward procedures are particular cases of this equation when $t = 1$ and $t = T$ respectively.

Question 2. Best state sequence

- Most likely path for a given observation O:

$$\begin{aligned}\operatorname*{argmax}_X P_\mu(X \mid O) &= \operatorname*{argmax}_X \frac{P_\mu(X, O)}{P_\mu(O)} \\ &= \operatorname*{argmax}_X P_\mu(X, O) \quad \text{(since } O \text{ is fixed)}\end{aligned}$$

- Compute the best sequence with the same recursive approach than in FB: Viterbi algorithm, $\mathcal{O}(TN^2)$.

- $\delta_j(t) = \max_{X_1 \ldots X_{t-1}} P_\mu(X_1 \ldots X_{t-1} s_j, o_1 \ldots o_t)$

  Highest probability of any sequence reaching state s_j at time t after emmitting $o_1 \ldots o_t$

- $\psi_j(t) = last(\operatorname*{argmax}_{X_1 \ldots X_{t-1}} P_\mu(X_1 \ldots X_{t-1} s_j, o_1 \ldots o_t))$

  Last state (X_{t-1}) in highest probability sequence reaching state s_j at time t after emmitting $o_1 \ldots o_t$

Viterbi algorithm

1. Initialization: $\forall j = 1 \dots N$
   $\delta_j(1) = \pi_j b_{jo_1}$
   $\psi_j(1) = 0$

2. Induction: $\forall t : 1 \leq t < T$
   $\delta_j(t+1) = \max_{1 \leq i \leq N} \delta_i(t) a_{ij} b_{jo_{t+1}} \quad \forall j = 1 \dots N$
   $\psi_j(t+1) = \operatorname{argmax}_{1 \leq i \leq N} \delta_i(t) a_{ij} \quad \forall j = 1 \dots N$

3. Termination: backwards path readout.
   - $\hat{X}_T = \operatorname{argmax}_{1 \leq i \leq N} \delta_i(T)$
   - $\hat{X}_t = \psi_{\hat{X}_{t+1}}(t+1)$
   - $P(\hat{X}) = \max_{1 \leq i \leq N} \delta_i(T)$

Question 3. Parameter Estimation

Obtain model parameters (A, B, π) for the model μ that maximizes the probability of given observation O:

$$(A, B, \pi) = \operatorname*{argmax}_{\mu} P_{\mu}(O)$$

Baum-Welch algorithm

- Baum-Welch algorithm (*aka* Forward-Backward):
  1. Start with an initial model μ_0 (uniform, random, MLE...)
  2. Compute observation probability (F&B computation) using current model μ.
  3. Use obtained probabilities as data to reestimate the model, computing $\hat{\mu}$
  4. Let $\mu = \hat{\mu}$ and repeat until no significant improvement.
- Iterative hill-climbing: Local maxima.
- Particular application of Expectation Maximization (EM) algorithm.
- EM Property: $P_{\hat{\mu}}(O) \geq P_{\mu}(O)$

Definitions

- $\gamma_i(t) = P_\mu(X_t = i \mid O) = \dfrac{P_\mu(X_t = i, O)}{P_\mu(O)} = \dfrac{\alpha_i(t)\beta_i(t)}{\sum_{k=1}^{N} \alpha_k(t)\beta_k(t)}$

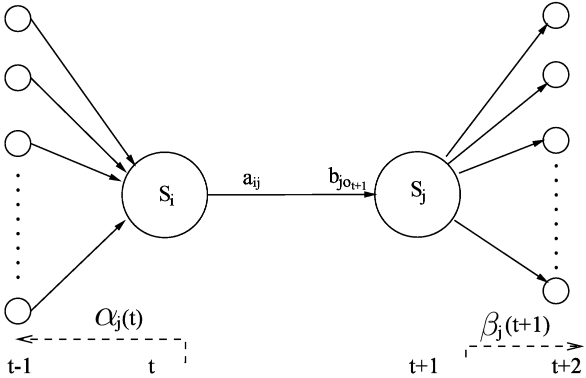
  Probability of being at state s_i at time t given observation O.

- $\varphi_t(i,j) = P_\mu(X_t = i, X_{t+1} = j \mid O) = \dfrac{P_\mu(X_t = i, X_{t+1} = j, O)}{P_\mu(O)}$

  $= \dfrac{\alpha_i(t) a_{ij} b_{jo_{t+1}} \beta_j(t+1)}{\sum_{k=1}^{N} \alpha_k(t)\beta_k(t)}$ probability of moving from state s_i at time t to state s_j at time $t+1$, given observation sequence O. Note that $\gamma_i(t) = \sum_{j=1}^{N} \varphi_t(i,j)$

$\sum_{t=1}^{T-1} \gamma_i(t)$ Expected number of transitions from state s_i in O.	$\sum_{t=1}^{T-1} \varphi_t(i,j)$ Expected number of transitions from state s_i to s_j in O.

Arc probability



Given an observation O, the model μ Probability $\varphi_t(i,j)$ of moving from state s_i at time t to state s_j at time $t+1$ given observation O.

Reestimation

Iterative reestimation

$$\hat{\pi}_i = \begin{array}{c}\text{Expected frequency in}\\ \text{state } s_i \text{ at time } (t=1)\end{array} = \gamma_i(1)$$

$$\hat{a}_{ij} = \frac{\begin{array}{c}\text{Expected number of}\\ \text{transitions from } s_i \text{ to } s_j\end{array}}{\begin{array}{c}\text{Expected number of}\\ \text{transitions from } s_i\end{array}} = \frac{\displaystyle\sum_{t=1}^{T-1} \varphi_t(i,j)}{\displaystyle\sum_{t=1}^{T-1} \gamma_i(t)}$$

$$\hat{b}_{jk} = \frac{\begin{array}{c}\text{Expected number of}\\ \text{emissions of } k \text{ from } s_j\end{array}}{\begin{array}{c}\text{Expected number}\\ \text{of visits to } s_j\end{array}} = \frac{\displaystyle\sum_{\substack{\{t:\, 1 \le t \le T,\\ o_t = k\}}} \gamma_t(j)}{\displaystyle\sum_{t=1}^{T} \gamma_t(j)}$$

1 Introduction

2 Statistical Models for NLP

3 Maximum Likelihood Estimation (MLE)

4 Maximum Entropy Modeling

5 Markovian Models

6 Similarity and Clustering
- Similarity
- Clustering
  - Hierarchical Clustering
  - Non-hierarchical Clustering

The Concept of Similarity

- *Similarity, proximity, affinity, distance, difference, divergence*
- We use *distance* when metric properties hold:
  - $d(x, x) = 0$
  - $d(x, y) \geq 0$ when $x \neq y$
  - $d(x, y) = d(y, x)$ (simmetry)
  - $d(x, z) \leq d(x, y) + d(y, z)$ (triangular inequation)
- We use *similarity* in the general case
  - Function: $sim : A \times B \rightarrow S$ (where S is often $[0, 1]$)
  - Homogeneous: $sim : A \times A \rightarrow S$ (e.g. word-to-word)
  - Heterogeneous: $sim : A \times B \rightarrow S$ (e.g. word-to-document)
  - Not necessarily symmetric, or holding triangular inequation.

The Concept of Similarity

- If A is a metric space, the distance in A may be used.
  - $D_{euclidean}(\vec{x}, \vec{y}) = |\vec{x} - \vec{y}| = \sqrt{\sum_i (x_i - y_i)^2}$
- Similarity *vs* distance
  - $sim_D(A, B) = \frac{1}{1+D(A,B)}$
  - monotonic: $min\{sim(x, y), sim(x, z)\} \geq sim(x, y \cup z)$

Applications

- Clustering, case-based reasoning, IR, ...
- Discovering related words - Distributional similarity
- Resolving syntactic ambiguity - Taxonomic similarity
- Resolving semantic ambiguity - Ontological similarity
- Acquiring selectional restrictions/preferences

Relevant Information

- Content (information about compared units)
  - Words: form, morphology, PoS, ...
  - Senses: synset, topic, domain, ...
  - Syntax: parse trees, syntactic roles, ...
  - Documents: words, collocations, NEs, ...
- Context (information about the situation in which simmilarity is computed)
  - Window–based *vs.* Syntactic–based
- External Knowledge
  - Monolingual/bilingual dictionaries, ontologies, corpora

Vectorial methods (1)

- L_1 norm, Manhattan distance, taxi-cab distance, city-block distance

$$L_1(\vec{x}, \vec{y}) = \sum_{i=1}^{N} |x_i - y_i|$$

- L_2 norm, Euclidean distance

$$L_2(\vec{x}, \vec{y}) = |\vec{x} - \vec{y}| = \sqrt{\sum_{i=1}^{N} (x_i - y_i)^2}$$

- Cosine distance

$$cos(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y}}{|\vec{x}| \cdot |\vec{y}|} = \frac{\sum_i x_i y_i}{\sqrt{\sum_i x_i^2} \cdot \sqrt{\sum_i y_i^2}}$$

Vectorial methods (2)

- L_1 and L_2 norms are particular cases of Minkowsky measure

$$D_{minkowsky}(\vec{x}, \vec{y}) = L_r(\vec{x}, \vec{y}) = \left(\sum_{i=1}^{N} (x_i - y_i)^r \right)^{\frac{1}{r}}$$

- Camberra distance

$$D_{camberra}(\vec{x}, \vec{y}) = \sum_{i=1}^{N} \frac{|x_i - y_i|}{|x_i + y_i|}$$

- Chebychev distance

$$D_{chebychev}(\vec{x}, \vec{y}) = \max_i |x_i - y_i|$$

Set-oriented methods (3): Binary–valued vectors seen as sets

- Dice. $S_{dice}(X,Y) = \dfrac{2 \cdot |X \cap Y|}{|X| + |Y|}$
- Jaccard. $S_{jaccard}(X,Y) = \dfrac{|X \cap Y|}{|X \cup Y|}$
- Overlap. $S_{overlap}(X,Y) = \dfrac{|X \cap Y|}{\min(|X|, |Y|)}$
- Cosine. $cos(X,Y) = \dfrac{|X \cap Y|}{\sqrt{|X| \cdot |Y|}}$

Above similarities are in $[0,1]$ and can be used as distances simply substracting: $D = 1 - S$

Set-oriented methods (4): Agreement contingency table

		Object i: 1	Object i: 0	
Object j	1	a	b	$a+b$
	0	c	d	$c+d$
		$a+c$	$b+d$	p

- Dice. $S_{dice}(X,Y) = \dfrac{2a}{2a+b+c}$
- Jaccard. $S_{jaccard}(X,Y) = \dfrac{a}{a+b+c}$
- Overlap. $S_{overlap}(X,Y) = \dfrac{a}{min(a+b,a+c)}$
- Cosine. $S_{overlap}(X,Y) = \dfrac{a}{\sqrt{(a+b)(a+c)}}$
- Matching coefficient. $S_{mc}(i,j) = \dfrac{a+d}{p}$

Distributional Similarity

- Particular case of vectorial representation where attributes are probability distributions

$$\vec{x}^T = [x_1 \ldots x_N] \text{ such that } \forall i, 0 \leq x_i \leq 1 \text{ and } \sum_{i=1}^{N} x_i = 1$$

- Kullback-Leibler Divergence (Relative Entropy)

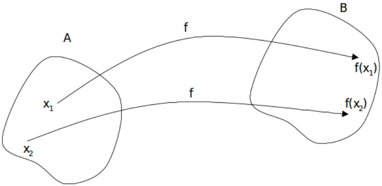
$$D(q||r) = \sum_{y \in Y} q(y) \log \frac{q(y)}{r(y)} \qquad \text{(non symmetrical)}$$

- Mutual Information

$$I(A, B) = D(h||f \cdot g) = \sum_{a \in A} \sum_{b \in B} h(a, b) \log \frac{h(a, b)}{f(a) \cdot g(b)}$$

(KL-divergence between joint and product distribution)

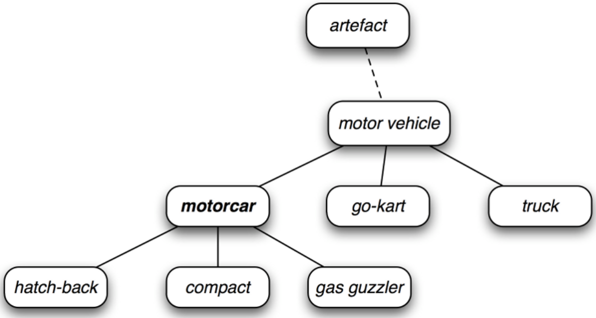
Semantic Similarity



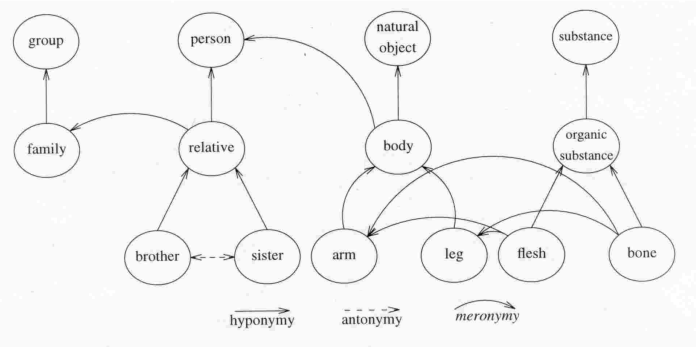
Project objects onto a semantic space:
$D_A(x_1, x_2) = D_B(f(x_1), f(x_2))$

- Semantic spaces: ontology (WordNet, CYC, SUMO, ...) or graph-like knowledge base (e.g. Wikipedia).
- Not easy to project words, since semantic space is composed of concepts, and a word may map to more than one concept.
- Not obvious how to compute distance in the semantic space.

WordNet


artefact
motor vehicle
motorcar
go-kart
truck
hatch-back
compact
gas guzzler

WordNet


group
person
natural object
substance
family
relative
body
organic substance
brother
sister
arm
leg
flesh
bone
hyponymy
antonymy
meronymy

Distances in WordNet

WordNet::Similarity

`http://maraca.d.umn.edu/cgi-bin/similarity/similarity.cgi`

Some definitions:

- $SLP(s_1, s_2)$ = Shortest Path Length from concept s_1 to s_2 (Which subset of arcs are used? antonymy, gloss, . . .)
- $depth(s)$ = Depth of concept s in the ontology
- $MaxDepth = \max_{s \in WN} depth(s)$
- $LCS(s_1, s_2)$ = Lowest Common Subsumer of s_1 and s_2
- $IC(s) = -log \frac{1}{P(s)}$ = Information Content of s (given a corpus)

Distances in WordNet

- Shortest Path Length: $D(s_1, s_2) = SLP(s_1, s_2)$
- Leacock & Chodorow: $D(s_1, s_2) = -log \frac{SLP(s_1, s_2)}{2 \cdot MaxDepth}$
- Wu & Palmer: $D(s_1, s_2) = \frac{2 \cdot depth(LCS(s_1, s_2))}{depth(s_1) + depth(s_2)}$
- Resnik: $D(s_1, s_2) = IC(LCS(s_1, s_2))$
- Jiang & Conrath:
  $D(s_1, s_2) = IC(s_1) + IC(s_2) - 2 \cdot IC(LCS(s_1, s_2))$
- Lin: $D(s_1, s_2) = \frac{2 \cdot IC(LCS(s_1, s_2))}{IC(s_1) + IC(s_2)}$
- Gloss overlap: Sum of squares of lengths of word overlaps between glosses
- Gloss vector: Cosine of second-order co-occurrence vectors of glosses

Distances in Wikipedia

- Measures using links, including measures used on WordNet, but applied to Wikipedia graph

  http://www.h-its.org/english/research/nlp/download/wikipediasimilarity.php

- Measures using content of articles (vector spaces)
- Measures using Wikipedia Categories

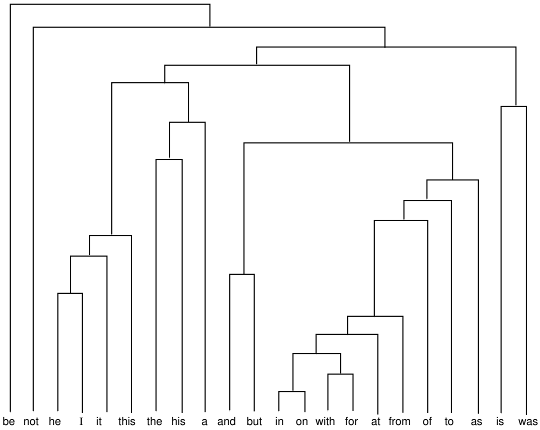
Clustering

- Partition a set of objects into clusters.
- Objects: features and values
- Similarity measure
- Utilities:
  - Exploratory Data Analysis (EDA).
  - Generalization (*learning*). Ex: *on Monday*, *on Sunday*, *? Friday*
- Supervised *vs* unsupervised classification
- Object assignment to clusters
  - Hard. *one cluster per object.*
  - Soft. *distribution* $P(c_i \mid x_j)$. *Degree of membership.*

Clustering

- Produced structures
  - Hierarchical (set of clusters + relationships)
    - Good for detailed data analysis
    - Provides more information
    - Less efficient
    - No single best algorithm
  - Flat / Non-hierarchical (set of clusters)
    - Preferable if efficiency is required or large data sets
    - K-means: Simple method, sufficient starting point.
    - K-means assumes euclidean space, if is not the case, EM may be used.
- Cluster representative
  - Centroid $\vec{\mu} = \frac{1}{|c|} \sum_{\vec{x} \in c} \vec{x}$

Dendogram



Single-link clustering of 22 frequent English words represented as a dendogram.

Hierarchical Clustering

- Bottom-up (Agglomerative Clustering)
  Start with individual objects, iteratively group the most similar.
- Top-down (Divisive Clustering)
  Start with all the objects, iteratively divide them maximizing within-group similarity.

Agglomerative Clustering (Bottom-up)

Input: A set $\mathcal{X} = \{x_1, \ldots, x_n\}$ of objects

A function sim: $\mathcal{P}(\mathcal{X}) \times \mathcal{P}(\mathcal{X}) \longrightarrow \mathcal{R}$

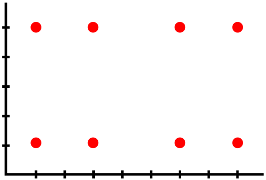
Output: A cluster hierarchy

for $i{:=}1$ **to** n **do** $c_i{:=}\{x_i\}$ **end**
$C{:=}\{c_1, \ldots, c_n\};\ \ j{:=}n+1$
while $C > 1$ **do**
$(c_{n_1}, c_{n_2}){:=}\arg\max_{(c_u,c_v)\in C\times C} \mathrm{sim}(c_u, c_v)$
$c_j = c_{n_1} \cup c_{n_2}$
$C{:=}C \setminus \{c_{n_1}, c_{n_2}\} \cup \{c_j\}$
$j{:=}j+1$
end–while

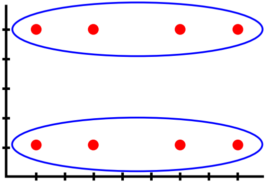
Cluster Similarity

- Single link: Similarity of two most similar members
  - Local coherence (close objects are in the same cluster)
  - Elongated clusters (chaining effect)
- Complete link: Similarity of two least similar members
  - Global coherence, avoids elongated clusters
  - Better (?) clusters
- UPGMA: Unweighted Pair Group Method with Arithmetic Mean
  - $\frac{1}{|X| \cdot |Y|} \sum_{x \in X} \sum_{y \in Y} D(x, y)$
  - Average pairwise similarity between members
  - Trade-off between global coherence and efficiency

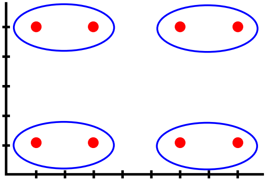
Examples



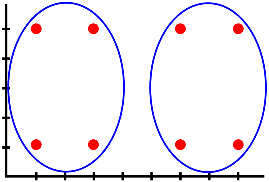
A cloud of points in a plane



Single-link clustering



Intermediate clustering



Complete-link clustering

Divisive Clustering (Top-down)

Input: A set $\mathcal{X} = \{x_1, \ldots, x_n\}$ of objects
A function coh: $\mathcal{P}(\mathcal{X}) \longrightarrow \mathcal{R}$
A function split: $\mathcal{P}(\mathcal{X}) \longrightarrow \mathcal{P}(\mathcal{X}) \times \mathcal{P}(\mathcal{X})$
Output: A cluster hierarchy

$C{:=}\{\mathcal{X}\}$; $c_1{:=}\mathcal{X}$; $j{:=}1$
while $\exists c_i \in C$ s.t. $|c_i| > 1$ **do**
    $c_u{:=}\arg\min_{c_v \in C} \text{coh}(c_v)$
    $(c_{j+1}, c_{j+2}) = \text{split}(c_u)$
    $C{:=}C \setminus \{c_u\} \cup \{c_{j+1}, c_{j+2}\}$
    $j{:=}j+2$
end–while

Top-down clustering

- Cluster splitting: Finding two sub-clusters
- Split clusters with lower *coherence*:
  - Single-link, Complete-link, Group-average
  - Splitting is a sub-clustering task:
    - Non-hierarchical clustering
    - Bottom-up clustering
- Example: Distributional noun clustering (Pereira et al., 93)
  - Clustering nouns with similar verb probability distributions
  - KL divergence as distance between distributions

$$D(p||q) = \sum_{x \in X} p(x) \log \frac{p(x)}{q(x)}$$

  - Bottom-up clustering not applicable due to some $q(x) = 0$

Non-hierarchical clustering

- Start with a partition based on random seeds
- Iteratively refine partition by means of *reallocating* objects
- Stop when cluster quality doesn't improve further
  - group-average similarity
  - mutual information between adjacent clusters
  - likelihood of data given cluster model
- Number of desired clusters ?
  - Testing different values
  - Minimum Description Length: the goodness function includes information about the number of clusters

K-means

- Clusters are represented by centers of mass (centroids) or a prototypical member (medoid)
- Euclidean distance
- Sensitive to outliers
- Hard clustering
- $\mathcal{O}(n)$

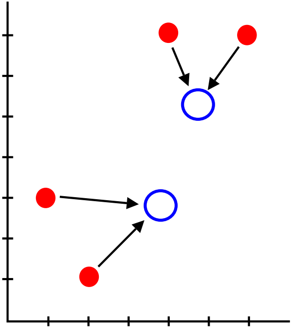
K-means algorithm

Input: A set $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \subseteq \mathcal{R}^m$
A distance measure $d : \mathcal{R}^m \times \mathcal{R}^m \longrightarrow \mathcal{R}$
A function for computing the mean $\mu : \mathcal{P}(\mathcal{R}) \longrightarrow \mathcal{R}^m$

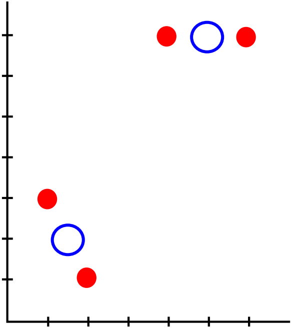
Output: A partition of $\mathcal{X}$ in clusters

Select k initial centers $\mathbf{f}_1, \ldots, \mathbf{f}_k$
while stopping criterion is not true **do**
  for all clusters c_j **do**
    $c_j := \{\mathbf{x}_i \mid \forall\, \mathbf{f}_l \;\; d(\mathbf{x}_i, \mathbf{f}_j) \leq d(\mathbf{x}_i, \mathbf{f}_l)\}$
  for all means $\mathbf{f}_j$ **do**
    $\mathbf{f}_j := \mu(c_j)$
end–while

K-means example



Assignment



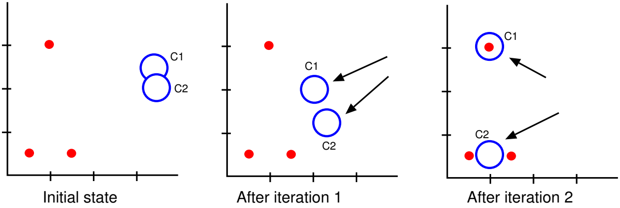
Recomputation of means

EM algorithm

- Estimate the (hidden) parameters of a model given the data
- Estimation–Maximization deadlock
  - Estimation: If we knew the parameters, we could compute the expected values of the hidden structure of the model.
  - Maximization: If we knew the expected values of the hidden structure of the model, we could compute the MLE of the parameters.
- NLP applications
  - Forward-Backward algorithm (Baum-Welch reestimation).
  - Inside-Outside algorithm.
  - Unsupervised WSD

EM example

- Can be seen as a *soft* version of K-means
- Random initial centroids
- Soft assignments
- Recompute (averaged) centroids



An example of using the EM algorithm for soft clustering

Clustering evaluation

- Related to a reference clustering: Purity and Inverse Purity.

$$P = \frac{1}{|D|}\sum_{c} \max_{x} |c \cap x|$$

$$IP = \frac{1}{|D|}\sum_{x} \max_{c} |c \cap x|$$

Where:
c = obtained clusters
x = expected clusters
$|D|$ = number of documents

- Without reference clustering: *Cluster quality* measures: Coherence, average internal distance, average external distance, etc.

References

- S. Abney, **Statistical Methods and Linguistics** In *The Balancing Act: Combining Symbolic and Statistical Approaches to Language*. The MIT Press, Cambridge, MA, 1996.
- L. Lee, **"I'm sorry Dave, I'm afraid I can't do that": Linguistics, Statistics, and Natural Language Processing**. National Research Council study on Fundamentals of Computer Science, 2003.
- T. Cover & J. Thomas, **Elements of Information Theory**. John Wiley & Sons, 1991.
- S.L. Lauritzen, **Graphical Models**. Oxford University Press, 1996
- C. Manning & H. Schütze, **Foundations of Statistical Natural Language Processing**. The MIT Press. Cambridge, MA. May 1999.

References

- D. Jurafsky & J.H. Martin. **Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics**, 2nd edition. Prentice-Hall, 2009.
- A. Berger, S.A. Della Pietra & V.J. Della Pietra, **A Maximum Entropy Approach to Natural Language Processing**. Computational Linguistics, 22(1):39-71, 1996.
- R Malouf, **A comparison of algorithms for maximum entropy parameter estimation**. In Proceedings of the Sixth Conference on Natural Language Learning (CoNLL-2002), Pages 49-55, 2002.
- L.R. Rabiner, **A tutorial on hidden Markov models and selected applications in speech recognition**. Proceedings of the IEEE, Vol. 77, num. 2, pg 257-286, 1989.
- A. Ratnaparkhi, **Maximum Entropy Models for Natural Language Ambiguity Resolution**. Ph.D Thesis. University of Pennsylvania, 1998.