Information Extraction and Question-Answering Systems

Foundations and methods

Dr. Günter Neumann
LT-Lab, DFKI
neumann@dfki.de

What the lecture will cover

- Machine Learning for IE
- Lexical processing
- Evaluation Methods
- Basic Terms & Examples
- Parsing of Unrestricted Text
- Generic NL Core system
- Domain Modelling
- Question/Answering Core components
- Advanced Topics

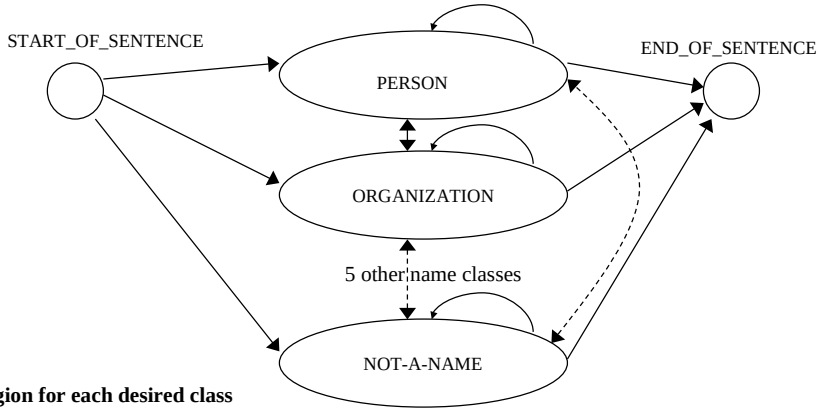
NE learning approaches

- Hidden Markov Models
- Maximum Entropy Modelling
- Decision tree learning

Hidden Markov Model for NE

- IdentiFinder™ developed at BBN
- View NE-task as a classification task
  - Every word is either part of some name
  - Or not a name
- Bigram language model for each name category
  - Predict the next category based on the previous word and previous name category
- HMM is language independent
  - Only simple word features for specific language
  - Evaluation performed for English & Spanish

Organize the states of the HMM into regions



- **One region for each desired class**
- **One for Not-A-Name**
- **Within each region, a model for computing the likelihood of words occuring within that region**

NE-based HMM

- Every word is represented by a state in the bigram model
- Associate a probability with every transition from the current word to the next word
- The likelihood of a sequence of words w_1 through w_n (a special +begin+ is used to compute the likelihood of w_1)

$$\prod_{i=1}^{n} p(w_i \mid w_{i-1})$$

NE-based HMM

- Find the most likely sequence of name classes (NC) given a word sequence W
  - ➢ max P(NC W)
  - ➢ Accordingly to Bayes´ Rule

$$P(NC \mid W) = \frac{P(NC) * (W \mid NC)}{P(W)} = \frac{P(W, NC)}{P(W)}$$

- Maximize the joint probability

Generation of words and name classes

1. Select a name-class NC, conditioning on the previous name-class and the previous word
2. Generate the first word inside the current name-class, conditioning on the current and previous name-class
3. Generate all subsequent words inside the current name-class, where each subsequent word is conditioned on its immediate predecessor
4. Repeat the 3 steps until an entire observed word sequence is generate

Example

Mr. Jones eats

Mr. <ENAMEX TYPE=PERSON> Jones </ENAMEX> eats

Possible (and hopefully most likely word-NC sequence):

P(Not-A-Name SOS, *+end+*) * P(*Mr.* Not-A-Name, SOS) *
P(*+end+* *Mr.* , Not-A-Name) * P(Person Not-A-Name, *Mr.*) *
P(*Jones* Person, Not-A-Name) * P(*+end+* *Jones*, Person) *
P(Not-A-Name Person,*Jones*) * P(*eats* Not-A-Name, Person) *
P(. *eats*,Not-A-Name) * P(*+end+* .,Not-A-Name) *
P(EOS Not-A-Name,.)

Word features <w,f> are the only language dependent part

- Easily determinable token properties:

Feature	Example	Intuition
fourDigitNum	1990	four digit year
containsDigitAndAlpha	A123-456	product code
containsCommaAndPeriod	1.00	monetary amount, percentage
otherNum	34567	other number
allCaps	BBN	Organisation
capPeriod	M.	Person name initial
firstWord	first word of sentence	ignore capitalization
initCap	Sally	capitalized word
lowerCase	can	uncapitalized word
other	,	punctuation, all other words

P(<anderson,initCap> <arthur, initCap>$_{-1}$, organization-name) >
P(<anderson,initCap> <arthur, initCap>$_{-1}$, person-name)

Top Level Model

- Probability for generating the first word of a name-class
  - Intuition: a word preceding the start of a NC (e.g., Mr.) and the word following a NC are strong indicators of the subsequent and preceding NC
  - Make a transition from one name-class to another
  - Calculate the likelihood of that word

$$P(NC \mid NC_{-1}, w_{-1}) * P(< w, f >_{first} \mid NC, NC_{-1})$$

P(Person Not-A-Name, Mr.) * P(Jones Person, Not-A-Name)

Top Level Model

- Generating all but the first word in a name-class

$$P(< w, f > \mid < w, f >_{-1}, NC)$$

- +end+ for the probability for any word to be the final word of its name-class

$$P(< +end+, other > \mid < w, f >_{final}, NC)$$

Training: estimating probabilities

- name-class bigram:

$$\Pr(NC \mid NC_{-1}, w_{-1}) = \frac{c(NC, NC_{-1}, w_{-1})}{c(NC_{-1}, w_{-1})}$$

- first-word-bigram:

$$\Pr(\langle w, f \rangle_{first} \mid NC, NC_{-1}) = \frac{c(\langle w, f \rangle_{first}, NC, NC_{-1})}{c(NC, NC_{-1})}$$

- non-first-word-bigram:

$$\Pr(\langle w, f \rangle \mid \langle w, f \rangle_{-1}, NC) = \frac{c(\langle w, f \rangle, \langle w, f \rangle_{-1}, NC)}{c(\langle w, f \rangle_{-1}, NC)}$$

where c(event) = #occurrences of event in training corpus

Handling of unknown words

- Vocabulary is built as it trains
- All unknown words are mapped to the token _UNK_
- _UNK_ can occur
  - As the current word, previous word, or both
- Train an unknown word model on held-out data
  - Gather statistics of unknown words in the midst of known words
- Approach in IdentiFinder
  - 50% hold out for unknown word model
  - Do the same for the other 50%
  - combine bigram counts for the first unknown training file

Back-off models

Models trained on hand-tagged corpus
=> Pr(X Y,Z) is not always available
=> fall back to weaker models:

Name-class bigram	**First-word bigram**	**Non-first-word bigram**
$P(NC \mid NC_{-1}, w_{-1})$	$P(< w, f >_{first} \mid NC, NC_{-1})$	$P(< w, f > \mid < w, f >_{-1}, NC)$
$P(NC \mid NC_{-1})$	$P(< w, f > \mid < +begin+, other >, NC)$	$P(< w, f > \mid NC)$
$P(NC)$	$P(< w, f > \mid NC)$	
	$P(w \mid NC) * P(f \mid NC)$	$P(w \mid NC) * P(f \mid NC)$
$\frac{1}{\#name-classes}$	$\frac{1}{\lvert V \rvert} * \frac{1}{\#name-classes}$	$\frac{1}{\lvert V \rvert} * \frac{1}{\#name-classes}$

Computing the weight

- Each back-off model is computed on the fly using P(X Y)*(1-λ), where

$$\lambda = \left(1 - \frac{old.c(Y)}{c(Y)}\right) * \frac{1}{1 + \frac{unique\_outcomes\_of\_Y}{c(Y)}}$$

- Old(Y): the sample size of the model from which backing-off is performed
- Using unique outcomes over the sample size: a crude measure of the certainty of the model

Results of Evaluation

- English (MUC-6, WSJ) and Spanish (MET-1): F-measure score

	Language	Best Result	IdentiFinder
Mixed Case	English	96.4	94.9
Upper Case	English	89	93.6
Speech form	English	74	90.7
Mixed Case	Spanish	93	90

On MUC-6 overall recall and precision: 96% R, 93% P

NLP task as classification problem

- Estimate probability that a class *a* appears with (or given) an event (context) *b*.
  - *P(a,b)*
  - *P(a b)*
- Maximum Likelihood Estimation
  - Corpus sparseness
  - Smoothing
  - Combining evidence
    - Independence assumptions
    - Interpolations
    - Etc.

Maximum Entropy Modelling

- An alternative estimation technique
- Able to deal with different kinds of evidence
- Maximum entropy method
  - Modell all that is known
  - Assume nothing about which is unknown
- Maximum Entropy (un-informative):
  - When one has no information to distinguish between the probability of two events, the best strategy is to consider them equally likely
  - Find the most uniform (maximum entropy) probability distribution that matches the observations

Entropy measures

- Entropy: a measure for the amount of uncertainty of a probability distribution. Shannon's entropy:

  $$H(p) = -\sum_i p_i \log p_i$$

- H reaches maximum, log(n), for p(x)=1/n
- H reaches minimum, 0, if one event e has p(e)=1, and the other 0.

Core idea of MEM

- Probability for a class Y and an object X depends solely on the *features* that are „active“ for the pair (X,Y)
- Features are the means through which an experimenter feeds problem-specific information
- The *importance* of each feature is determined automatically by running a parameter estimation algorithm over pre-classified set of examples („training-set“)
- Advantage: experimenter need only tell the model *what* information to use, since the model will automatically determine *how* to use it.

Maximum Entropy Modeling

- Random process
  - produces an output value *y*, a member from a finite set *Y*
  - Might be influenced by some contextual information *x*, a member from a finite set *X*
- Construct a stochastic model that accurately describes the random process
  - Estimate the conditional probability *P(Y X)*
- Training data: $(x_1, y_1) , (x_2, y_2) , ..., (x_N, y_N)$

$$r(x, y) \equiv \frac{c(x, y)}{N}$$

Simple example

- Task: estimate a joint probability distribution *p* defined over {x,y}×{0,1}
- Known facts (constraints) about *p*
  - p(x,0)+p(y,0)=0.6
  - p(x,0)+p(y,0)+p(x,1)+p(y,1)=1

P(a,b)	0	1	
X	?	?	
Y	?	?	
Total	.6		1

One way to satisfy constraints

P(a,b)	0	1	
X	.5	.1	
Y	.1	.3	
Total	.6		1

Is this also the most accurate one?

Simple Example

- Observed facts are constraints for the desired model *p*
- Observed fact p(x,0)+p(y,0)=0.6 is implemented as a constraint of *feature* f_1 of model *p*, $E_p f_1$, where

$$E_p f_1 = \sum_{a \in \{x,y\}, b \in \{0,1\}} p(a,b) f_1(a,b) \qquad f_1(a,b) = \begin{cases} 1 \text{ if } b = 0 \\ 0 \text{ otherwise} \end{cases}$$

Most *uncertain* way to satisfy constraints:

P(a,b)	0	1	
X	.3	.2	
Y	.3	.2	
Total	.6	.4	1

Histories, binary features & futures

- History b: information derivable from the corpus relative to a token:
  - text window around token w_i, e.g. $w_{i-2},\ldots,w_{i+2}$
  - word features of these tokens
  - POS, other complex features
- Features:
  - yes/no-questions on history used by models to determine probabilities of
- Futures: what we are predicting (e.g., POS, name classes)

Features represent evidence

- a = what we are predicting (e.g., tags)
- b = what we observe (e.g., words)
- A feature f has the form

  $f_{y,q}(a,b)=1$ if a=y & q(b) = true
  0 otherwise
- E.g.,

  $f_{NNP,q1}(a,b)=1$ if a=NNP & q1(b) = true
  $f_{VBG,q2}(a,b)=1$ if a=VBG & q2(b) = true

Weight features with conditional probability model

$$P(a \mid b) = \frac{\prod_{j=1}^{k} \alpha_j^{f_j(a,b)}}{Z(b)} = \frac{\prod_{j=1}^{k} \alpha_j^{f_j(a,b)}}{\sum_a \prod_{j=1}^{k} \alpha_j^{f_j(a,b)}}$$

- Z(b) = normalization factor
- $\alpha_j > 0$: weights for feature f_j
- P(a b): (normalized) product of weights of active feature on the (a,b) pair, i.e., those features f_j such that $f_j(a,b)=1$

Maximum Likelihood Estimation

- Given a model form, choose parameters to maximize likelihood of training data
- r(a,b): observed probability of (a,b) in training data
- $Q=\{p\ p(a\ b)=(1/Z(b))\Pi_{j=1\ldots k}\alpha_j^{fj(a,b)}\}$
- $L(p)=\Sigma_{a,b} r(a,b) \log p(a\ b)$
- $p_{ML}=\text{argmax}_{p\in Q} L(p)$

Principle of Maximum Entropy

- Use the probability distribution that has maximum entropy, or that is maximally uncertain, from those that are consistent with observed evidence
- P = { models consistent with evidence}
- H(p)= entropy of p
- $p_{ME}=\text{argmax}_{p\in P}H(p)$

The Conditional Maximum Entropy Framework

(Berger et al., Computational Linguistics, Vol 22, No 1, 1996)

- $E_r f_j$=observed expectation of f_j
  $= \Sigma_{a,b} r(a,b)\, f_j(a,b)$
- $E_p f_j$=model's expectation of f_j
  $= \Sigma_{a,b} r(b)p(a\ b)\, f_j(a,b)$
- $P=\{p \ \ E_p f_j = E_r f_j, j=1...k\}$
- $H(p)= -\Sigma_{a,b} r(b)p(a\ b)\log p(a\ b)$
  - Conditional entropy for p(a b)
- $p_{ME}=\text{argmax}_{p\in P}H(p)$

Duality of ME and ML

- By maxent criterion, p_{ME} *must* have form $p_{ME}(a\ b)=(1/Z(b))\Pi_{j=1\ldots k}\alpha_j^{fj(a,b)}$
- ME and ML solutions are the same
  - $p_{ME}(a\ b)= p_{ML}(a\ b)$
  - ML: form is assumed without justification
  - ME: constraints on feature expectations are assumed, form is derived

ME/ML Parameter Estimation

- Generalized Iterative Scaling (Darroch&Ratcliff, 72)
  - Goal: computation of the alphas
  - Requires, that for each event (a,b) the number of features that are active equals some constant C
  - If not true find constant C and correction feature f_{k+1}
    - $f_{k+1}(a,b)=C- \Sigma_{j=1\ldots k} f_j(a,b)$, $C=\max \Sigma_{j=1} f_i(x)$
  - Iterative updates
    - $\alpha_j^{(0)} =1$
    - $\alpha_j^{(n)} = \alpha_j^{(n-1)} (E_r f_j / E_p^{(n-1)} f_j)^{1/C}$
- Improved Iterative Scaling (Della Pietra et al.,97)
  - Does not require correction feature

Advantage of Maxent

- Diverse forms of evidence
- No independence assumptions: contrast with naive bayes
- Feature weights are determined automatically
- No smoothing

How to specify a maxent model

- Outcomes: What are we predicting
- Questions: What information is useful for predicting?
  Both determine set of candidate features F:
  F={$f_{y,q}$ y is outcome, q a question}
- Feature selection: Given candidate feature set F, what subset of it do we actually use?

IE-related MEM
Introductory Example
(Diploma thesis by Volker Morbach)

- Example:

<FN 2><FN 1>Die Apollinaris & Schweppes GmbH & Co.**</FN></FN>** (Bad Neuenahr) will kuenftig rund 60 bis 70 Prozent ihrer Getraenke per Bahn transportierten. **<GR 1>**Der Umsatz**</GR> <TZ 1>**stieg**</TZ> <BT 1>**auf 367,9 (1993: 348,1) Millionen DM**</BT>**, **<GR 2>**der Ueberschuss**</GR> <TZ 2>**erhoehte sich**</TZ>** **<BT 2>**auf 44,7 (30,9) Millionen DM**</BT>** .

IE-related MEM
Introductory Example

- Example Event (1):
  - Prediction = FN, Context: cl1cl2P(gmbh)cr1cr2

SEM	FN	FN	Pred.	FN	FN
TC	Other Symbol	First capital	Mixed word, First capital	Other Symbol	Lowercase word
POS			NOUN		
STEM			gmbh		
TOKEN	&	schweppes	gmbh	&	co.

IE-related MEM Introductory Example

- Example Event (2):
  - Prediction = GR, Context:

SEM	*N*	GR		TZ	BT
TC	Separator Symbol	First capital	First capital	Lowercase word	Lowercase word
POS	INTP	DEF	NOUN	VERB	PREP
STEM	.	d-det	umsatz	steig	auf
TOKEN	.	der	umsatz	stieg	auf

IE-related MEM Introductory Example

- Example Features:
  - From Example (1): *Good* feature:

    If (a==FN && STEM[0]="gmbh") then

    return 1.0
  - From Example (2): *Bad* feature:

    If (a==GR && TC[2]="Lowercase Word") then

    return 1.0

IE-related MEM Model Training

- There are two widely used algorithms for training maxent models:
  - GIS (Generalized Iterative Scaling)
    - Good: Not Numerically fragile
    - Bad: Needs the existence of a correction feature
  - IIS (Improved Iterative Scaling)
    - Good: No correction feature necessary
    - Good: Faster
    - Bad: Numerically fragile

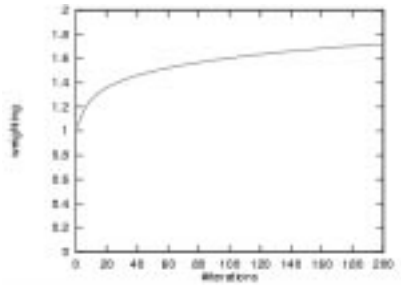
IE-related MEM Model Training

- Whatever algorithm is used, in each case model training means computing feature weights αj. The first iteration starts with every $\alpha j=1.0$. Subsequencing iterations will change this value: either to a value greater than 1.0 (if the corresponding feature is considered as *good*) or to a value less than 1.0 (but greater than 0.0) (if the corresponding feature is considered as *bad*).

IE-related MEM Model Training

- Example (1): *Good* feature:

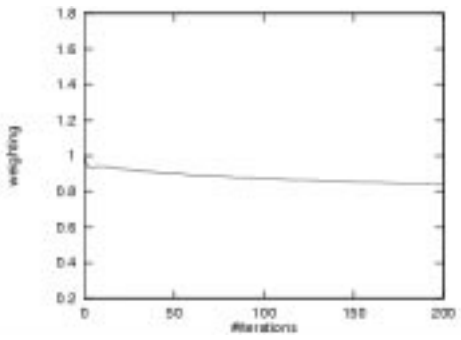
If (a==FN && STEM[0]="gmbh") then return 1.0



IE-related MEM Model Training

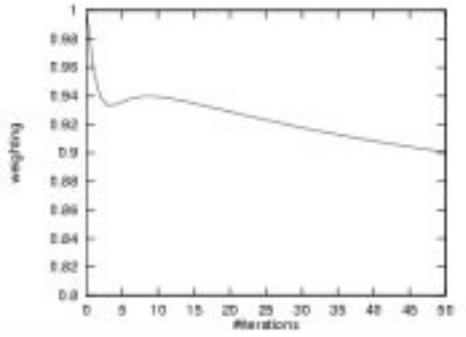
- Example (2): *Bad* feature:

If (a==GR && TC[2]="Lowercase Word") then return 1.0



IE-related MEM Model Training

- Note, that (as shown by example 2) computation is not monotonic. Here, we depict an enlarged version of the not-monotonic area:



Maximum Entropy Named Entity (MENE, Borthwick,99)

- Uses Maxent as „black-box“ tool
- Allows use of broad range of knowledge sources
- State-of-art accuracy
- Trans-lingual portability (English version adapted to Japanese)

Knowledge representation

- Outcomes:
  - N tags for NE (MUC-7 classes)
  - For each particular class x

    x_start,x_continue,x_end,x_unique

    [Jerry Lee Lewis flew to Paris]

    [pers_start,pers_continue,pers_end,other,other,loc_unique]
  - 4n+1 tags

Types of features

- Binary
  - Token properties which are either on or a off for a given token (e.g., All-caps, 2-digit-number,only-digits,initial-cap)
  - Overlapping allowed (in contrast to IdentiFinder), i.e., no ordering presupposed
- Lexical
  - Lexical lookup for words in the context for a current token
  - Lexicon is build automatically (just build a vocabulary *V* as „all words *w*: c(*w*) > 2
  - More elaborate methods possible

Types of features

- Dictionaries
  - Multi-word entries of pre-classified NE words (e.g, first names)
  - Ambiguites handled because of overlapping properties (Maxent will find out weighting)
  - However, some possible dictionaries are rejected because of decreased performance (e.g., location identifiers, world airlines)
- Reference resolution
  - Similar to SMES system
  - Substring match

Feature selection

1. Put all possible features from the classes to be included into the model into a *feature pool*
   1. Lexical features for range $w_{-2}...w_2$, vocabulary size of V, then (5*(V+1)*29) lexical features
2. Select all features which fire at least three times on the training corpus
3. Features which predict the tag *other* have to fire six times to be included
4. Lexical features which activate on w_{-2} and w_2 are excluded if they predict *other*

Evaluation

- Results for MUC-7
  - 93%P, 85%R, 88,80% F
  - Fourth best system
- Upper case results
  - MENE: 77.98% F
  - MENE-Proteus: 82.76% F
- Evaluation for Japanese (MET-2)
  - 83.80 % F

Decision Tree Learning

- A decision tree takes as input a situation described by a set of attributes and returns a yes/no "decision".
- A decision tree can
  - represent any discrete-valued function (or more specifically, any propositional or Boolean function),
  - be rewritten in disjunctive normal form (DNF).
- ID3 (and its extended version C4.5) are widely used alorithms developed by Ross Quinlan, informally performing:
  - If there exists N classes, what is the best (minimal) set of questions/attributes (selected from a finite set of attributes) I have to answer/determine values in order to classify an object X

Basic idea

- We are given a set of records, each a number of attribute/value pairs.
- One of these attributes represents the category of the record. The problem is to determine a decision tree that on the basis of answers to questions about the non-category attributes predicts correctly the value of the category attribute.
- Usually the category attribute takes only the values {true, false}, or {success, failure}, or something equivalent. In any case, one of its values will mean failure.

Golf playing example

We are dealing with records reporting on weather conditions for playing golf. The categorical attribute specifies whether or not to play.

Data structure

ATTRIBUTE	POSSIBLE VALUES
Outlook O	sunny, overcast, rain
Temperature T	continuous
Humidity H	Continuous
Windy W	true, false

Training data

O	T	H	W	PLAY
sunny	85	85	false	Don't Play
sunny	80	90	true	Don't Play
overcast	83	78	false	Play
rain	70	96	false	Play
rain	68	80	false	Play
rain	65	70	true	Don't Play
overcast	64	65	true	Play
sunny	72	95	false	Don't Play
sunny	69	70	false	Play
rain	75	80	false	Play
sunny	75	70	true	Play
overcast	72	90	true	Play
overcast	81	75	false	Play
rain	71	80	true	Don't Play

The basic ideas behind ID3

- In the decision tree each node corresponds to a non-categorical attribute and each arc to a possible value of that attribute. A leaf of the tree specifies the expected value of the categorical attribute for the records described by the path from the root to that leaf. [This defines what is a Decision Tree.]
- In the decision tree at each node should be associated the non-categorical attribute which is most informative among the attributes not yet considered in the path from the root. [This establishes what is a "Good" decision tree.]
- Entropy is used to measure how informative is a node. [This defines what we mean by "Good". By the way, we already used this notion when introducing MEM.]

Basic Decision Tree Algorithm

Recursively build a decision tree top-down through batch processing of the training data.

***DTree*(examples, attributes):**
If all examples are in one category, return a leaf node with this category as a label.
Else if attributes are empty then return a leaf node labelled with the category which is most common in examples.
Else Pick an attribute, A, for the root. **(use attribute with largest gain)**
For each possible value v_i for A
Let $examples_i$ be the subset of examples that have value v_i for A.
Add a branch out of the root for the test $A= v_i$.
If $examples_i$ is empty
Then create a leaf node labelled with the category which is most common in examples
Else recursively create a subtree by calling *DTree*($examples_i$, attributes - {A})

Entropy and Information Gain

- For a given a probability distribution P = (p_1, p_2, .., p_n) the information conveyed by this distribution, also called the *Entropy* of P, is:

$$H(p) = -\sum_i p_i \log_2 p_i$$

  - For example, if P is (0.5, 0.5) then H(P) is 1, if P is (0.67, 0.33) then H(P) is 0.92, if P is (1, 0) then H(P) is 0

- If a set T of records is partitioned into disjoint exhaustive classes C_1, C_2, .., C_k on the basis of the value of the categorical attribute, then the information needed to identify the class of an element of T is Info(T) = H(P), where P = ($|C_1|/|T|$, $|C_2|/|T|$, ..., $|C_k|/|T|$)
- In our weather example, we have Info(T) = H(9/14, 5/14) = 0.94

Continued

- If we first partition T on the basis of the value of a non-categorical attribute X into sets T_1, T_2, .., T_n then the information needed to identify the class of an element of T becomes the weighted average of the information needed to identify the class of an element of T_i, i.e. the weighted average of Info(T_i):

$$\text{Info}(X,T) = \sum_{i=1}^{n} \frac{|T_i|}{|T|} * \text{Info}(T_i)$$

Example:

Info(O,T) = 5/14*H(2/5,3/5) + 4/14*H(4/4,0) + 5/14*H(3/5,2/5)

= 0.694

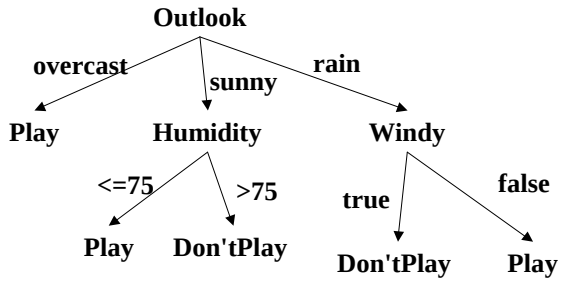
Information gain

- Gain(X,T) = Info(T) - Info(X,T):
  The difference between the information needed to identify an element of T and the information needed to identify an element of T after the value of attribute X has been obtained, that is, this is the gain in information due to attribute X.
- Example, gain of
  - Outlook attribute:
    Gain(O,T) = Info(T) - Info(O,T) = 0.94 - 0.694 = 0.246
  - Windy attribute:
    Info(W,T)=0.892 and Gain(W,T)=0.048.
  - Thus Outlook offers a greater informational gain than Windy.
- Use gain for ranking attributes and to build decision trees where at each node is located the attribute with greatest gain among the attributes not yet considered in the path from the root.

Benefits of Information Gain

- Use gain for ranking attributes and to build decision trees where at each node is located the attribute with greatest gain among the attributes not yet considered in the path from the root.
- The intent of this ordering are twofold:
  - To create small decision trees so that records can be identified after only a few questions.
  - To match a hoped for minimality of the process represented by the records being considered(Occam's Razor).
- In general, finding a minimal decision tree consistent with a set of data is NP-hard.
- The simple recursive algorithm does a greedy heuristic search for a fairly simple tree but cannot guarantee optimal.

Decision tree for golfing example



Using gain ratios

- Gain tends to favor attributes that have a large number of values. E.g., if we have an attribute D that has a distinct value for each record, then Info(D,T) is 0, thus Gain(D,T) is maximal. To compensate for this Quinlan suggests using the following ratio instead of Gain:

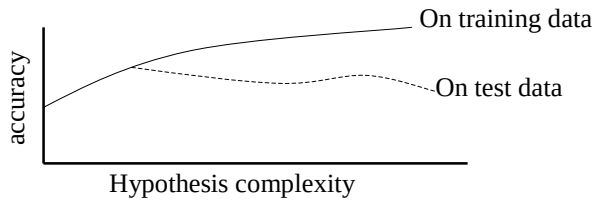
$$\text{GainRatio}(D,T) = \frac{\text{Gain}(D,T)}{\text{SplitInfo}(D,T)}$$

- SplitInfo(D,T) is the information due to the split of T on the basis of the value of the goal attribute D. Thus SplitInfo(D,T) is $H(|T_1|/|T|, |T_2|/|T|, .., |T_m|/|T|)$ where $\{T_1, T_2, .. T_m\}$ is the partition of T induced by the value of D

- Example for SplitInfo(Outlook,T)
  - -5/14*log(5/14) - 4/14*log(4/14) - 5/14*log(5/14) = 1.577
  - GainRatio of Outlook is 0.246/1.577 = 0.156.

Overfitting and Pruning

- Learning a tree that classifies the training data perfectly may not lead to the tree with the best generalization performance since
  - There may be noise in the training data that the tree is fitting.
  - The algorithm might be making some decisions toward the leaves of the tree that are based on very little data and may not reflect reliable trends in the data.
- A hypothesis, h, is said to overfit the training data if there exists another hypothesis, h', such that h has smaller error than h' on the training data but h' has smaller error on the test data than h.



Methods to avoid overfitting

- Two basic approaches to when pruning occurs
  - Prepruning: Stop growing the tree at some point during construction when it is determined that there is not enough data to make reliable choices.
  - Postpruning: Grow the full tree and then remove nodes that seem do not have sufficient evidence.
- Methods for evaluating which subtrees to prune:
  - Cross-validation: Reserve some of the training data as a hold-out set (validation set, tuning set) to evaluate utility of subtrees.
  - Statistical testing: Perform some statistical test on the training data to determine if any observed regularity can be dismissed as likely due to random chance.
  - Minimum Description Length (MDL): Determine if the additional complexity of the hypothesis is less complex than just explicitly remembering any exceptions.

Reduced-Error Pruning

- A post-pruning, cross-validation approach:
  Partition training data into ``grow'' and ``validation''sets.
  Build a complete tree for the ``grow'' data.
  Until accuracy on validation set decreases do:
  For each non-leaf node, n, in the constructed tree
  Temporarily prune the tree below n and replace it with a leaf labelled with the majority category.
  Test the accuracy of the resulting pruned tree on the validation set.
  Permanently prune the node that results in the greatest increase in accuracy on the validation set.

- Major problem is that it reduces the amount of data used to construct a tree, which can be very damaging if relatively little training data is available.
- If the algorithm can take a parameter setting that determines the complexity of the hypothesis it will build (i.e. number of nodes). A good value for this parameter can be determined using cross-validation and then the system retrained on the entire training set using this value.

Missing attribute values (C4.5)

- In building a decision tree we can deal with training sets that have records with unknown attribute values by evaluating the gain, or the gain ratio, for an attribute by considering only the records where that attribute is defined.
- Classify records that have unknown attribute values by estimating the probability of the various possible results. In our golfing example, if we are given a new record for which the outlook is sunny and the humidity is unknown, we proceed as follows:
  - We move from the Outlook root node to the Humidity node following the arc labeled 'sunny'. At that point since we do not know the value of Humidity we observe that if the humidity is at most 75 there are two records where one plays, and if the humidity is over 75 there are three records where one does not play. Thus one can give as answer for the record the probabilities (0.4, 0.6) to play or not to play.

NER based on decision tree learning

(Gallipi, Coling 96)

- Goal: select and organize features into a discrimination tree, one tree for each type of NE
- Features:
  - POS
  - Designator („Corp“, „GmbH“, ...)
  - Morphology (Ending, Word length, ...)
  - Word lists (Person, companies)
  - Templates (<NNP CN_design>)

Hybrid system by A. Gallippi

- Hand-built phrasal templates for delimitation (proper noun, ampersand, hypen, comma, ...)
- Separate DT for each name class
- Step 1: delimit proper nouns
- Step 2: to classify a PN
  - Compute features for window around PN
  - Compute weight for each name class using its DT
  - Merge results to choose a name class

Recognition steps: Delimitation and classification

- Delimitation is the determination of the boundaries of the NE, while classification serves to provide a more specific category
  - Orginal: JohnSmith, chairman of Safetek, announced his resignation yesterday.
  - Delimit: <NE>JohnSmith </NE>, chairman of <NE> Safetek </NE>, announced his resignation yesterday.
  - Cassify: <PN>JohnSmith </PN>, chairman of <CN> Safetek </CN>, announced his resignation yesterday.

Delimitation

- Application of phrasal templates
- Built by hand using logical operators to combine features strongly assdociated with NE
  - Proper noun
  - Ampersand, hyphen, comma

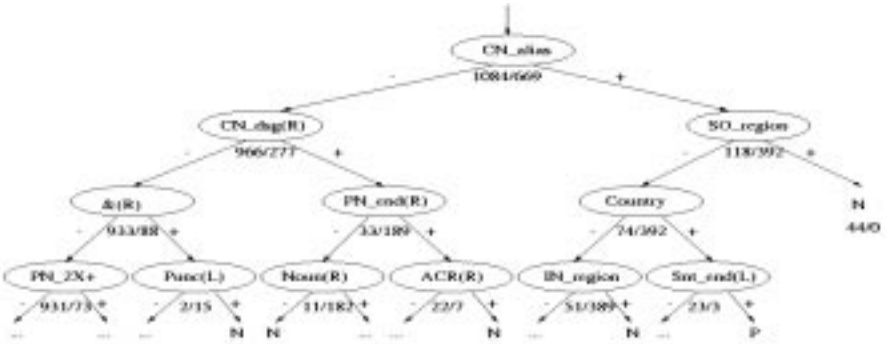
Decision trees for learning classification knowledge

- Starting point: each word is tagged with all of its associated features
- Features are obtained through automated and manual techniques
- Decision tree is then constructed from the initial feature set using a recursive partitioning algorithm (ID3)

Features

Type	Feature	Example
POS	Proper Noun Common Noun	Aristotle philosophy
Disgnator	Company Person Location Date	Corp.,Ltd Mr. President Country, State, City Month,Day of weel
Morphology	Capitalization Company suffix Word length	A-, B- -corp, -tee WL>8,WL<3
List	Companies Persons Keywords	IBM, AT&T Smith, Michael Based in, said he
Template	Company Person Location Date Proper Name	NNP CN_descr P_desig NNP NNP L_desig MM Num, Num NNP NNP
Special purpose	LCS Duplicated PNs	VW <- Volkswagen DUP_2+

Decision Trees generated for companies



- Context level of tree is 3: the feature in question must occur within the region starting 3 words to the left and ending 3 words to the right of the proper name‘s left boundary
- (L/R) indicates that the feature must appear to left/right of left boundary of proper noun
- Numbers represent numbers of negative/positive examples from training corpus

Cross-language porting

software requirements:

- tokenizer (non-trivial for non-token languages, e.g. Japanese)
- word feature identification
- POS tagger etc.

needed data:

- annotated training texts in new language
- translated dictionary (word lists)

Evaluation

- English:
  - Types: companies, persons, locations, dates
  - F=94 % (weighted average)
  - Strongest features for English

Feature	Companies	Persons	Locations
F1	CAP	P_desig	CAP
F2	CN_desig	CAP	L_desig
F3	CN_alias	ATH_reg	In
F4	Hyphen	F_I_L	Region

ATH_reg: occurs in Author tags
In: lexical „in“
Region: geographical region name
F_I_L: First name+initial+last name

Evaluation (cont.)

- Spanish
  - F=89.2 % (weighted average)
    - Date: 100%, Loc:88.6, Pers: 87.4,Com:81.6
- System adaptations
  - Specific decision trees are generated from the feature set optimized for English and applied to Spanish text
  - ... Minor adjustments made to the feature set in order to improve Spanish

Type	Feature	Example
List	Companies Keywords	IBM, AT&T „del“ (OF THE)
Template	Person Person Date Date	FN DE LN FN DE NNP Num OF MM Num OF MM OF Num

Evaluation (cont.)

- Japanese
  - F=83.1 % (weighted average)

    Date: 92.3%, Loc:81.3, Pers: 85.7,Com:60.0
- System adaptation
  - Same as for Spanish
  - Specialized Japanese tokenizer
  - Pre-tagged Japanese text