CS114 Lecture 20
Classifiers

April 9, 2014

Professor Meteer

Thanks to Mitch Marcus, Chris Manning & Massimo Poesio, and Lance Ramshaw (and others from BBN) for slides

Assgn 4: Classification

- Train the NLKT Naive Bayes classifier on features from the Switchboard DIalog Act Corpus
- Classify the utterances in a test set according to the dialog act tag (See assignment)
- Documentation for the corpus in
  - http://compprag.christopherpotts.net/swda.html
- You need to first pull out a test set (~25% of the whole corpus)

Assgn 4: Classification

- Tests to run
  1. Compute the baseline using just the words as features.
  2. Compute the "next level" using bigrams and trigrams
  3. Design 3-5 more features and see how much you can improve the performance. You can use anything in the data as features. Be creative
- Submit
  - code
  - Results:
    - accuracy
    - confusion matrix for each feature set
  - discussion of the features including why you chose that feature and what it contributed to the results

Classifiers

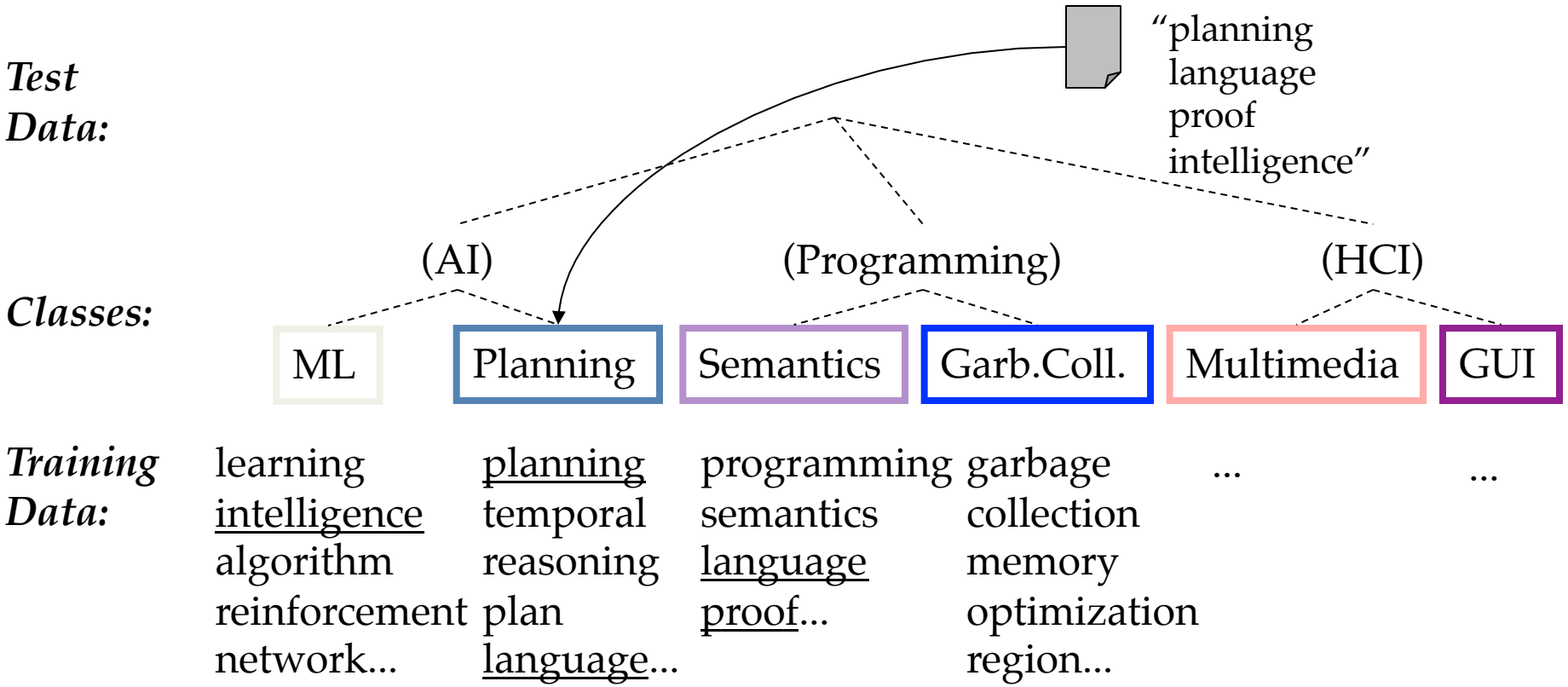
- Problem of classification
  - Task of choosing the correct class label for a given input
  - Each input generally determined independently
  - Set of labels defined in advance
- Type of classifiers
  - Decision Trees
  - Naïve Bayes Classifiers
  - Maximum Entropy Classifiers

Categorization/Classification

- Given:
  - A description of an instance, $x \in X$, where X is the *instance language* or *instance space*.
    - Issue: how to represent text documents.
  - A fixed set of categories:

    $C = \{c_1, c_2, \ldots, c_n\}$
- Determine:
  - The category of x: $c(x) \in C$, where $c(x)$ is a *categorization function* whose domain is X and whose range is C.
    - We want to know how to build categorization functions ("classifiers")

Document Classification



EXAMPLES OF TEXT CATEGORIZATION

- LABELS=BINARY
  - "spam" / "not spam"
- LABELS=TOPICS
  - "finance" / "sports" / "asia"
- LABELS=OPINION
  - "like" / "hate" / "neutral"
- LABELS=AUTHOR
  - "Shakespeare" / "Marlowe" / "Ben Jonson"
  - The Federalist papers
- LABELS=Intent/task
  - "get_balance" / "pay_bill" / "transfer_funds"

Methods (1)

- Manual classification
  - Used by Yahoo!, Looksmart, about.com, ODP, Medline
  - very accurate when job is done by experts
  - consistent when the problem size and team is small
  - difficult and expensive to scale
- Automatic document classification
  - Hand-coded rule-based systems
    - Reuters, CIA, Verity, …
    - Commercial systems have complex query languages

Methods (2)

- Supervised learning of document-label assignment function: Autonomy, Kana, MSN, Verity, …
  - *Naive Bayes (simple, common method)*
  - k-Nearest Neighbors (simple, powerful)
  - Support-vector machines (new, more powerful)
  - … plus many other methods
  - No free lunch: requires hand-classified training data
  - But can be built (and refined) by amateurs

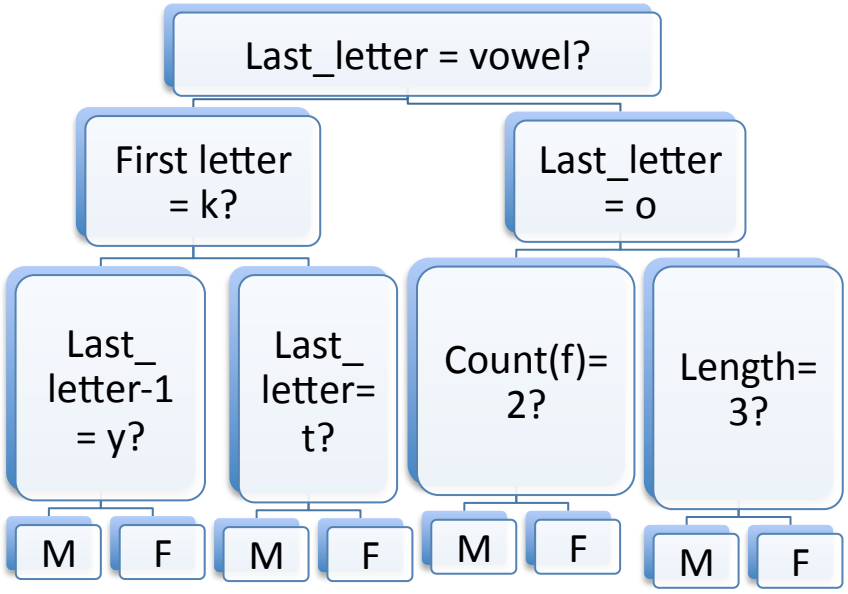
Decision Trees

- Decision tree parts
  - Decision nodes: check feature values, one per node
  - Leaf nodes: Assign labels
  - Root: Initial decision
  - Decision stump: Decision tree with a single node which splits the data based on that one feature
- Operationally
  - Input: Element to be classified + feature vector with values
  - Traverse the tree, at each node check the feature value to determine which path to take
  - The label at the leaf is the class of the input

Decision tree for name gender task

Yes: go left

No: go right



Building a decision tree

- Simplest
  - Build decision stumps for each feature
  - Pick the best based on results (accuracy)
  - Start with that, split the data and repeat
- Using Entropy and information gain
  - Calculate the entropy of the labels in the corpus
    - (e.g. M, F, M, F, F, F, M, F, F,)
    - Sum of the probability of each label times the log probability of that label
    - $H = -\Sigma_{l \,|in|\, labels} P(l) \times log_2 P(l)$
  - Calculate the entropy for each of the decision stump's leaves
  - Information gain =
    - Original entropy minus the average of those leaf entropy values (weighted by the number of samples in each leaf)
  - *Higher the information gain, the better job that stump (e.g. feature) does at dividing the input*

Problems with decision trees

- Each node divides the data
  - Lower nodes may "overfit" training
    - Reflect idiosyncrasies of the training data
  - Can prune back
- Features have to be checked in a particular order
  - Some features may need to repeated in multiple branches of the tree
  - Weak features may not get a chance to apply
    - too low in the tree and not enough data

Calculating Entropy on a List of Labels in Python

```
import math
def entropy(labels):
freqdist = nltk.FreqDist(labels)
probs = [freqdist.freq(l) for l in nltk.FreqDist(labels)]
return -sum([p * math.log(p,2) for p in probs])
```

```
>>> print entropy(['male', 'male', 'male', 'male'])
0.0
>>> print entropy(['male', 'female', 'male', 'male'])
0.811278124459
>>> print entropy(['female', 'male', 'female', 'male'])
1.0
>>> print entropy(['female', 'female', 'male', 'female'])
0.811278124459
>>> print entropy(['female', 'female', 'female', 'female'])
0.0
```

Bayesian Methods

- Learning and classification methods based on probability theory (see spelling / POS)
- Bayes theorem plays a critical role
- Build a *generative model* that approximates how data is produced
- Uses *prior* probability of each category given no information about an item.
- Categorization produces a *posterior* probability distribution over the possible categories given a description of an item.

Bayes' Rule once more

C: Set classes
X: set of feature-value vectors describing the target

$$P(C, X) = P(C \mid X)P(X) = P(X \mid C)P(C)$$

$$\boxed{P(C \mid X) = \frac{P(X \mid C)P(C)}{P(X)}}$$

Maximum a posteriori Hypothesis

H: Set of hypotheses (classes)
h: hypothesis
MAP: maximum a posteriori
D: Description (feature-values)

$$h_{MAP} \equiv \underset{h \in H}{\operatorname{argmax}} P(h \mid D)$$

$$= \underset{h \in H}{\operatorname{argmax}} \frac{P(D \mid h)P(h)}{P(D)}$$

$$= \underset{h \in H}{\operatorname{argmax}} P(D \mid h)P(h)$$

As *P(D)* is constant

Maximum likelihood Hypothesis

- If all hypotheses are a priori equally likely, we only need to consider the P(D|h) term:

$$h_{ML} \equiv \underset{h \in H}{\operatorname{argmax}} P(D \mid h)$$

Naive Bayes Classifiers

Task: Classify a new instance *D* based on a tuple of attribute value into one of the classes $c_j \in C$

$$D = \langle x_1, x_2, \ldots, x_n \rangle$$

$$c_{MAP} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j \mid x_1, x_2, \ldots, x_n)$$

$$= \underset{c_j \in C}{\operatorname{argmax}} \frac{P(x_1, x_2, \ldots, x_n \mid c_j) P(c_j)}{P(x_1, x_2, \ldots, x_n)}$$

$$= \underset{c_j \in C}{\operatorname{argmax}} P(x_1, x_2, \ldots, x_n \mid c_j) P(c_j)$$

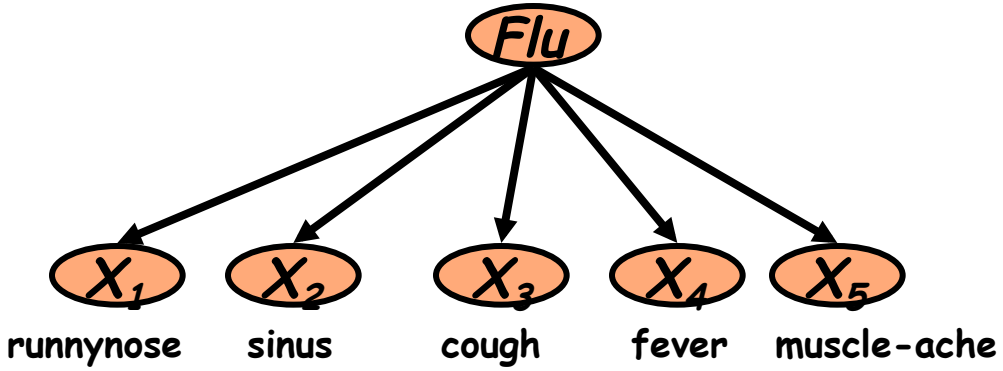
Naïve Bayes Classifier: Assumption

- $P(c_j)$
  - Can be estimated from the frequency of classes in the training examples
- $P(x_1, x_2, \ldots, x_n | c_j)$
  - O($|X|^n \bullet |C|$) parameters
  - Could only be estimated if a very, very large number of training examples was available.

Naïve Bayes Conditional Independence Assumption:

- Assume that the probability of observing the conjunction of attributes is equal to the product of the individual probabilities $P(x_i | c_j)$.

The Naïve Bayes Classifier

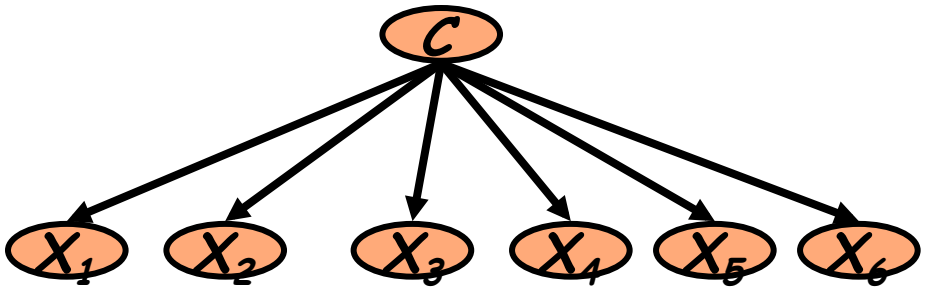


- Conditional Independence Assumption: features are independent of each other given the class:

$$P(X_1,\ldots,X_5 \mid C) = P(X_1 \mid C) \bullet P(X_2 \mid C) \bullet \cdots \bullet P(X_5 \mid C)$$

- This model is appropriate for binary variables

Learning the Model



- First attempt: maximum likelihood estimates
  - simply use the frequencies in the data (+ smoothing, of course…)

$$\hat{P}(c_j) = \frac{N(C = c_j)}{N}$$

Count of class c_j

Number of items

$$\hat{P}(x_i \mid c_j) = \frac{N(X_i = x_i, C = c_j)}{N(C = c_j)}$$

Using Naive Bayes Classifiers to Classify Text: Basic method

- Attributes are text positions, values are words.

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_i P(x_i \mid c_j)$$

$$= \underset{c_j \in C}{\operatorname{argmax}} P(c_j) P(x_1 = \text{"our"} \mid c_j) \cdots P(x_n = \text{"text"} \mid c_j)$$

  - **Still too many possibilities**
  - **Assume that classification is *independent* of the positions of the words**
    - Use same parameters for each position
    - Result is *bag of words* model

Text Classification Algorithms: Learning

- From training corpus, extract Vocabulary
- Calculate required $P(c_j)$ and $P(x_k \mid c_j)$ terms
  - For each c_j in C do

$docs_j \leftarrow$ subset of documents for which the target class is c_j

=

$$P(c_j) \leftarrow \frac{|docs_j|}{|\text{total \# documents}|}$$

$Text_j \leftarrow$ single document containing all $docs_j$

For each word x_k in *Vocabulary*

$n_k \leftarrow$ number of occurrences of x_k in $Text_j$

$$P(x_k \mid c_j) \leftarrow \frac{n_k}{n} \qquad \textit{(must be smoothed)}$$

Naïve Bayes: Classifying

- positions ← all word positions in current document which contain tokens found in Vocabulary
- Return c_{NB}, where

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_{i \in positions} P(x_i \mid c_j)$$

Underflow Prevention

- Multiplying lots of probabilities, which are between 0 and 1 by definition, can result in floating-point underflow.
- Since log(*xy*) = log(*x*) + log(*y*), it is better to perform all computations by summing logs of probabilities rather than multiplying probabilities.

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} \log P(c_j) + \sum_{i \in positions} \log P(x_i \mid c_j)$$

- Class with highest final un-normalized log probability score is still the most probable.

Feature selection via Mutual Information

- We might not want to use all words, but just reliable, good discriminating terms
- In training set, choose k words which best discriminate the categories.
- One way is using terms with maximal Mutual Information with the classes:

$$I(w,c) = \sum_{e_w \in \{0,1\}} \sum_{e_c \in \{0,1\}} p(e_w, e_c) \log \frac{p(e_w, e_c)}{p(e_w)p(e_c)}$$

  - For each word w and each category c

Feature selection via MI (contd.)

- For each category we build a list of *k* most discriminating terms.
- For example (on 20 Newsgroups):
  - ***sci.electronics:*** circuit, voltage, amp, ground, copy, battery, electronics, cooling, …
  - ***rec.autos:*** car, cars, engine, ford, dealer, mustang, oil, collision, autos, tires, toyota, …
- Greedy: does not account for correlations between terms

Feature Selection

- Mutual Information
  - Clear information-theoretic interpretation
  - May select rare uninformative terms
- Commonest terms:
  - No particular foundation
  - In practice often is 90% as good
- Other methods: Chi-square, etc.…
- Modern methods use *regularization*
  - Removes redundant and irrelevant features

PANTEL AND LIN: SPAMCOP

- Uses a Naïve Bayes classifier
- M is spam if P(Spam|M) > P(NonSpam|M)
- Method
  - Tokenize message using Porter Stemmer
  - Estimate P(W|C) using m-estimate (a form of smoothing)
  - Remove words that do not satisfy certain conditions
  - Train: 160 spams, 466 non-spams
  - Test: 277 spams, 346 non-spams
- Results: ERROR RATE of 4.33%
  - Worse results using trigrams

Naive Bayes is Not So Naive

- Naïve Bayes: First and Second place in KDD-CUP 97 competition, among 16 (then) state of the art algorithms (Knowledge Discovery and Data Mining)

  Goal: Financial services industry direct mail response prediction model: Predict if the recipient of mail will actually respond to the advertisement – 750,000 records.

- Robust to Irrelevant Features

  Irrelevant Features cancel each other without affecting results

  Instead Decision Trees can heavily suffer from this.

- Very good in Domains with many equally important features

  Decision Trees suffer from *fragmentation* in such cases – especially if little data

- A good dependable baseline for text classification (but not the best)!
- Optimal if the Independence Assumptions hold: If assumed independence is correct, then it is the Bayes Optimal Classifier for problem
- Very Fast: Learning with one pass over the data; testing linear in the number of attributes, and document collection size
- Low Storage requirements

Maxiumum Entropy Classifiers

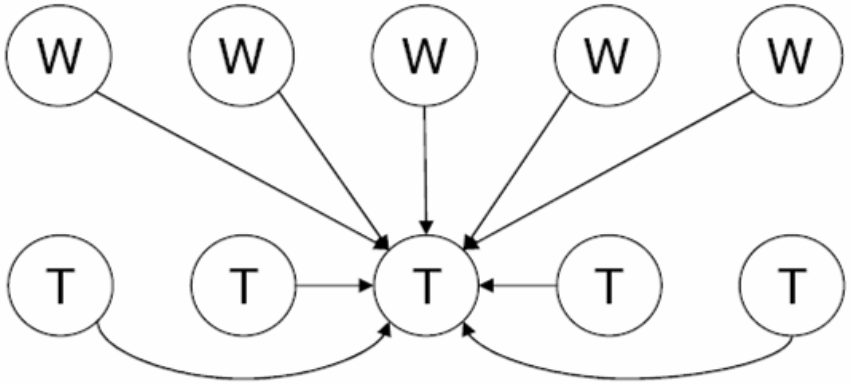
- Looks for a set of parameters that maximizes the total likelihood of the training corpus

  $P(features) = \Sigma_{x \,|in|\, corpus}\, P(label(x)|features(x))$

  $P(label|features) = P(label, features) / \Sigma_{label}\, P(label, features)$

- Iterative optimization techniques to calculate model
  - initialize the model's parameters to random values
  - then repeatedly refine those parameters to bring them closer to the optimal solution
  - Not guaranteed to be optimal (hill climbing)

MaxEnt Name Finding



- **Predicts tags given words**
- **Features can be based on arbitrary sets of context elements.**
  - With Conditional Random Fields, they can also depend on other tags.
- **Features can overlap with each other in arbitrary ways.**

BBN TECHNOLOGIES

Learning Feature Weights

- Each feature is true of ("activated for") some fraction of the examples in the training set
- Training the model involves learning a set of feature weights that correctly predict the actual training set activation levels for each feature.
- The correct weights for each feature are highly dependent on the other features.

BBN TECHNOLOGIES

MaxEnt as a Log Linear Model

- **Start with training data vectors X_i with answers Y_i**
- **Define a set of (typically boolean) features f_i (x,y)**
  - E.g. age>60, weight >250, and diabetes=true
  - Features can overlap with each other
- **The algorithm learns feature weights w_i**
- **P(y|x) is computed as:**

$$p(y \mid x) = \frac{1}{Z_w(x)} \exp\left(\sum_i w_i f_i(x, y) \right)$$

- **Where the normalizing constant Z is:**

$$Z_w(x) = \sum_y \exp\left(\sum_i w_i f_i(x, y) \right)$$

- **Effectively, we sum the weights for features that apply and exponentiate to get a score; then normalize the scores for all possible outcomes to get probabilities.**

Computing the Feature Weights

- **Various methods can be used to find W*, the weight vector that maximizes the likelihood of the training.**
  - (Since the likelihood function is smooth and convex in W.)
- **Iterative Scaling is one out of date but simple method**
  - Start with w_i=0 for all *i*
  - Update each w_i then roughly as follows:

$$\Delta w_i = \log \frac{\overline{p}(f_i)}{p_w(f_i)}$$

  - Namely increase the weight if the measured frequency of the feature is greater than the model's prediction, and vice versa.
  - Repeat until the weights stabilize.
- **Gradient ascent or conjugate gradient can also be used.**

Review: Applications

- Information Extraction
- Summary generation
- Machine Translation
- Document organization/classification
- Automatic indexing of books
- Improve Internet search results (location Clinton/South Carolina vs. President Clinton)

NP Chunking as tagging

[$_{NP}$ Pierre Vinken $_{NP}$] , [$_{NP}$ 61 years $_{NP}$] old , [$_{VP}$ will join $_{VP}$] [$_{NP}$ the board $_{NP}$] of [$_{NP}$ directors $_{NP}$] as [$_{NP}$ a non-executive director $_{NP}$] [$_{NP}$ Nov 29 $_{NP}$]

Pierre/B Vinken/I ,/O 61/B years/I old/O ,/O will/O join/O the/B board/I of/O directors/B as/O a/B non-executive/I director/I Nov/B 29/I ./O

I	Inside chunk
O	Outside chunk
B	Begin chunk

Memory-Based XP Chunker

Assigning non-recursive phrase brackets (Base XPs) to phrases in context: (XP → (Specifier) X (Complement))

[NP The	woman NP]	[VP will	give VP]	[NP Mary NP]	[NP a	book NP]	.
Det	NN	MD	VB	NNP	Det	NN	.
I-NP	I-NP	I-VP	I-VP	I-NP	B-NP	I-NP	

Convert NP, VP, ADJP, ADVP, PrepP, and PP brackets to classification decisions (I/O/B tags) (Ramshaw & Marcus, 1995).

Features:

POS_{-2}, $IOBtag_{-2}$, $word_{-2}$,
POS_{-1}, $IOBtag_{-1}$, $word_{-1}$,
POS_{focus}, $word_{focus}$,
POS_{+1},
$word_{+1}$, POS_{+2}, $word_{+2}$, → IOB tag

Memory-Based XP Chunker

- Results (WSJ corpus)

type	prec	recall	F1
NP	92.5	92.2	92.3
VP	91.9	91.7	91.8
ADJP	68.4	65.0	66.7
ADVP	78.0	77.9	77.9
Prep	95.5	96.7	96.1
PP	91.9	92.2	92.0
ADVFunc	78.0	69.5	73.5

- One-pass segmentation and chunking for all XP
- Useful for: Information Retrieval, Information Extraction, Terminology Discovery, etc.

Review: Named Entities

The who, where, when & how much in a sentence

The task: identify atomic elements of information in text

- person names
- company/organization names
- locations
- dates&times
- percentages
- monetary amounts

Extraction Example

– **George Garrick, 40 years old, president of the London-based European Information Services Inc., was appointed chief executive officer of Nielsen Marketing Research, USA.**

George Garrick, 40 years old,

Nielsen Marketing Research, USA.

Position	Company	Location	Person	Status
President	European Information Services, Inc.	London	George Garrick	Out
CEO	Nielsen Marketing Research	USA	George Garrick	In

Machine Learning Approaches

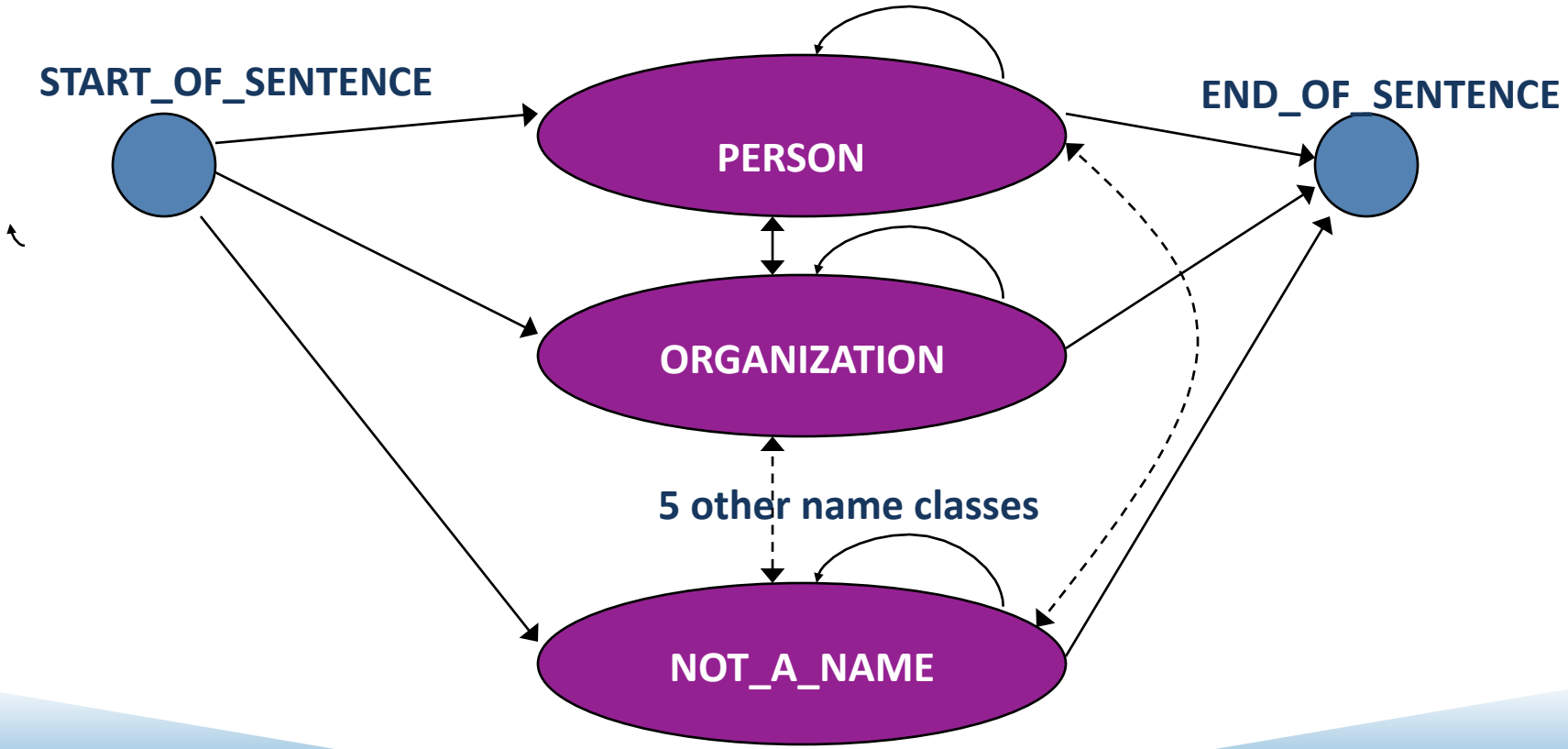
- ML approaches frequently break down the NE task into two parts:
  1. ***Find*** entity boundaries
  2. *Classify* entities into NE categories

- Or: Reduce NE boundary *detection and classification* to IOB tagging
  - O – outside, B-XXX – first word in NE, I-XXX – all other words in NE
  - Argentina B-LOC
    played O
    with O
    Del B-PER
    Bosque I-PER

IdentiFinder (Nymble) [Bikel et al 99]

- Based on Hidden Markov Models
- Features
  - Capitalisation
  - Numeric symbols
  - Punctuation marks
  - Position in the sentence
  - 14 features in total, combining above info, e.g., containsDigitAndDash (*e.g. 09-96*), containsDigitAndComma (*e.g. 23,000.00*)

Nymble's structure (simplified)



IdentiFinder (2)

- MUC-6 (English) and MET-1(Spanish) corpora used for evaluation
- Mixed case English
  - IdentiFinder - 94.9% f-measure
  - Best rule-based – 96.4%
- Spanish mixed case
  - IdentiFinder – 90%
  - Best rule-based - 93%
  - Lower case names, noisy training data, less training data
- Training data: 650,000 words, but similar performance with half of the data. Less than 100,000 words reduce the performance to below 90% on English

Named Entity From J&M

- Named Entity Recognition as Sequence Labeling
- Word by word
- Class indicates both boundary and type

Words	Label
American	B_{ORG}
Airlines	I_{ORG}
,	O
a	O
unit	O
of	O
AMR	B_{ORG}
Corp.	I_{ORG}
,	O
immediately	O
matched	O
the	O
move	O
,	O
spokesman	O
Tim	B_{PERS}
Wagner	I_{PERS}
said	O
.	O

Types and Features

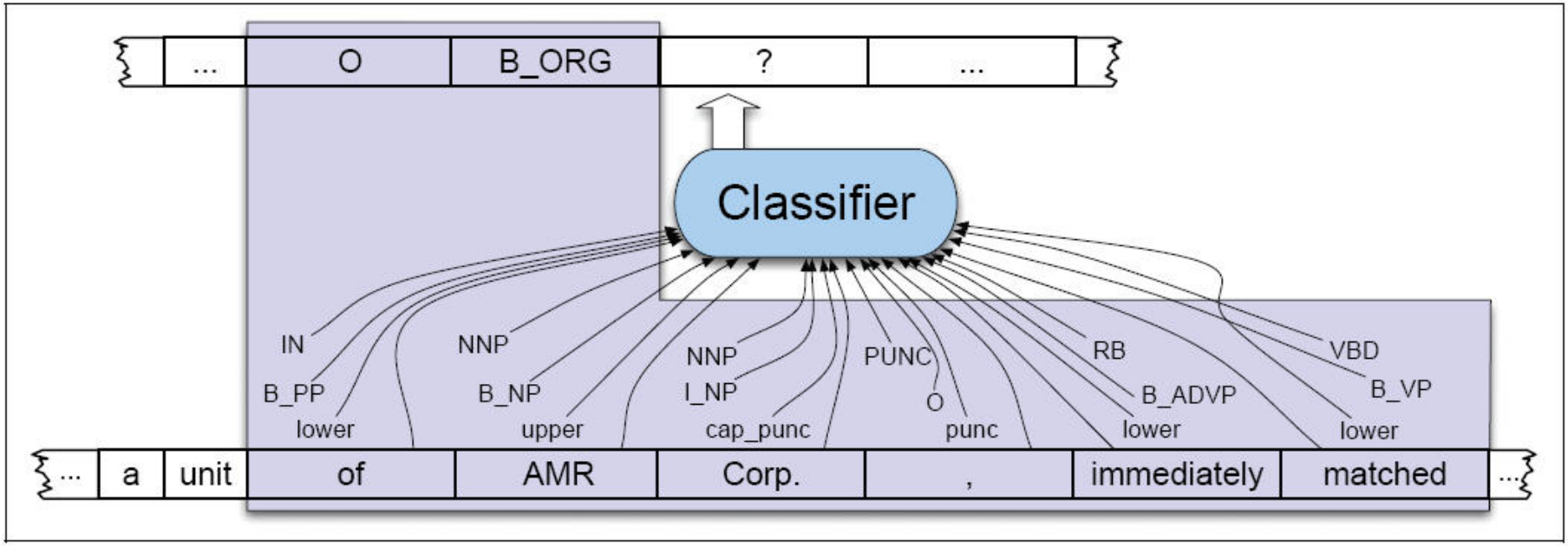
Type	Example
People	*Turing* is often considered to be the father of modern computer science.
Organization	The *IPCC* said it is likely that future tropical cyclones will become more intense.
Location	The *Mt. Sanitas* loop hike begins at the base of *Sunshine Canyon*.
Geo-Political Entity	*Palo Alto* is looking at raising the fees for parking in the University Avenue district.
Facility	Drivers were advised to consider either the *Tappan Zee Bridge* or the *Lincoln Tunnel*.
Vehicles	The updated *Mini Cooper* retains its charm and agility.

Feature	Explanation
Lexical items	The token to be labeled
Stemmed lexical items	Stemmed version of the target token
Shape	The orthographic pattern of the target word
Character affixes	Character-level affixes of the target and surrounding words
Part of speech	Part of speech of the word
Syntactic chunk labels	Base-phrase chunk label
Gazetteer or name list	Presence of the word in one or more named entity lists
Predictive token(s)	Presence of predictive words in surrounding text
Bag of words/Bag of N-grams	Words and/or *N*-grams occurring in the surrounding context

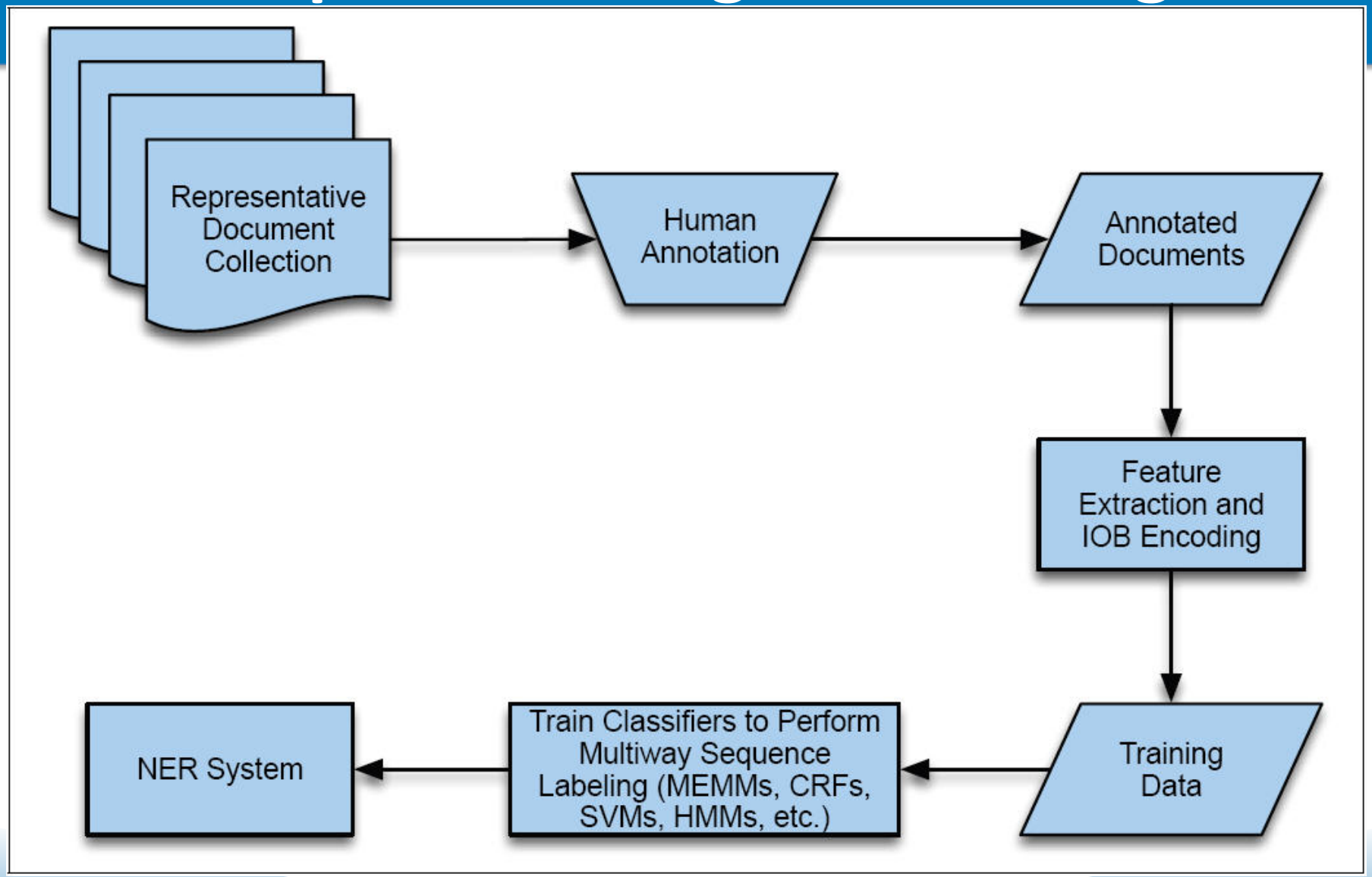
Sample word by word feature extraction and labeling for NER

Features				Label
American	NNP	B_{NP}	cap	B_{ORG}
Airlines	NNPS	I_{NP}	cap	I_{ORG}
,	PUNC	O	punc	O
a	DT	B_{NP}	lower	O
unit	NN	I_{NP}	lower	O
of	IN	B_{PP}	lower	O
AMR	NNP	B_{NP}	upper	B_{ORG}
Corp.	NNP	I_{NP}	cap_punc	I_{ORG}
,	PUNC	O	punc	O
immediately	RB	B_{ADVP}	lower	O
matched	VBD	B_{VP}	lower	O
the	DT	B_{NP}	lower	O
move	NN	I_{NP}	lower	O
,	PUNC	O	punc	O
spokesman	NN	B_{NP}	lower	O
Tim	NNP	I_{NP}	cap	B_{PER}
Wagner	NNP	I_{NP}	cap	I_{PER}
said	VBD	B_{VP}	lower	O
.	PUNC	O	punc	O

NER Classifier



Steps in labeling and training



Entropy

- Entropy(self-information)

$$H(p) = H(X) = -\sum_{x \in \chi} p(x) \log_2 p(x)$$

  - the amount of information in a random variable
  - average uncertainty of a random variable
  - the average length of the message needed to transmit an outcome of that variable
  - the size of the search space consisting of the possible values of a random variable and its associated probabilities
- Properties
  - $H(X) \geq 0$ ($H(X) = 0$: providing no new information)
  - increases with message length

Entropy Example

- Simplified Polynesian
  - letter frequencies

i	p	t	k	a	i	u
P(i)	1/8	1/4	1/8	1/4	1/8	1/8

  - per-letter entropy

$$H(P) = -\sum_{i \in \{p,t,k,a,i,u\}} P(i) \log P(i) = 2.5 \text{ bits}$$

  - Coding

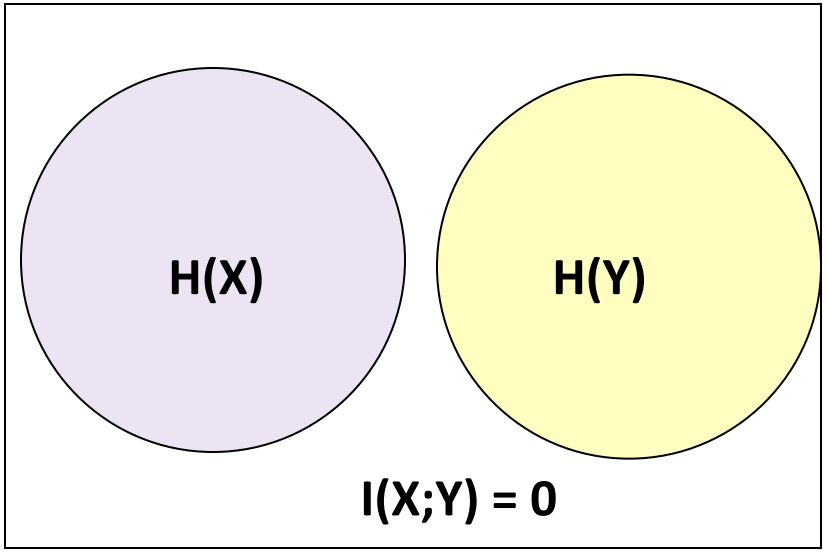
p	t	k	a	i	u
100	00	101	01	110	111

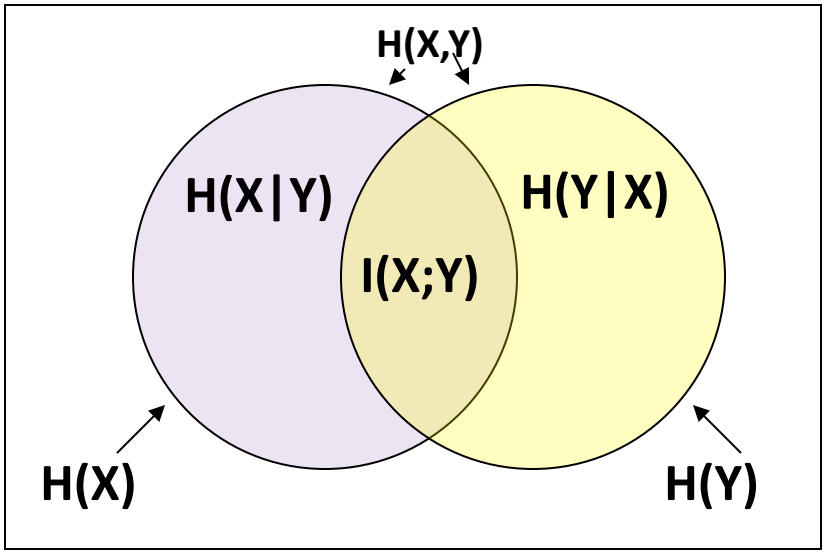
From Manning & Schutze, 1999

Perplexity and Entropy

- Both measure the (un)certainty of a model
  - How many choices are there at any given point
- Perplexity is $2^{(\text{entropy})}$ that is $2^{H(x1n,m)}$
  - Manning & Schutze hypothesize speech people want to show bigger gains when they reduce perplexity
  - Lowering perplexity from 940 to 540 is more impressive then reducing cross entropy from 9.9 to 9.1 bits

Mutual information and entropy





- I(X;Y) is 0 iff two variables are independent
- For two dependent variables, mutual information grows not only with the degree of dependence, but also according to the entropy of the variables

From Manning & Schutze, 1999

Feature selection via Mutual Information

- Problem: From training set of documents for some given class (topic), choose k words which best discriminate that topic.
- One way is using terms with maximal Mutual Information with the classes:

$$I(w,c) = \sum_{e_w \in \{0,1\}} \sum_{e_c \in \{0,1\}} p(e_w, e_c) \log \frac{p(e_w, e_c)}{p(e_w)p(e_c)}$$

  – For each word w and each category c

Joint entropy and conditional entropy

- Simplified Polynesian revisited
  - All words of consist of sequence of CV (consonant-vowel) syllables

Marginal probabilities (per-syllable basis)

	p	t	k	
a	**1/16**	6/16	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	1

Per-letter basis probabilities

p	t	k	a	i	u
1/16	3/8	1/16	1/4	1/8	1/8

double

Feature Selection: An Example

- Test Corpus:
  - Reuters document set.
    - Words in corpus: 704903
  - Sample subcorpus of ten documents with word "cancer"
    - Words in subcorpus: 5519

- "cancer" occurs:
  - 181 times in subcorpus
  - 201 times in entire document set

Most probable words given that "Cancer" appears in the document

311 the	56 said
181 cancer	54 for
171 of	37 on
141 and	36 about
137 in	35 but
123 a	35 are
106 to	34 it
71 women	33 have
69 is	33 at
65 that	32 they
64 s	30 with
61 breast	29 who

Words sorted by I(w,`cancer`)

Word	#c	total	$2^{I(w,'cancer')}$	P(w|'cancer')	P(w)
Lung	15	15	128	0.00272	2.13e-05
Cancers	14	14	128	0.00254	1.99e-05
Counseling	14	14	128	0.00254	1.99e-05
Mammograms	11	11	128	0.00199	1.56e-05
Oestrogen	10	10	128	0.00181	1.42e-05
Brca	8	8	128	0.00145	1.13e-05
Brewster	9	9	128	0.00163	1.28e-05
Detection	7	7	128	0.00127	9.93e-06
Ovarian	7	7	128	0.00127	9.93e-06
Incidence	6	6	128	0.00109	8.51e-06
Klausner	6	6	128	0.00109	8.51e-06
Lerman	6	6	128	0.00109	8.51e-06
Mammography	4	4	128	0.000725	5.67e-06