Evaluation: The 2×2 contingency matrix

System	Actual target	Actual ¬ target
selected	tp	fp
¬selected	fn	tn

Accuracy = $(tp + tn)/N$, where $N = tp + tn + fp + fn$

Error = $(fn + fp)/N = 1 -$ Accuracy

Why is this a bad measure for many NLP tasks?

Precision and recall

- Precision is defined as a measure of the proportion of selected items that the system got right:

$$\text{precision} = \frac{tp}{tp + fp}$$

- Recall is defined as the proportion of the target items that the system selected:

$$\text{recall} = \frac{tp}{tp + fn}$$

These two measures allow us to distinguish between excluding target items and returning irrelevant items.

They still require human-made "gold standard" judgements.

The F measure

Weighted harmonic mean: The F measure (where $F = 1 - E$):

$$F = \frac{1}{\alpha\frac{1}{P} + (1-\alpha)\frac{1}{R}}$$

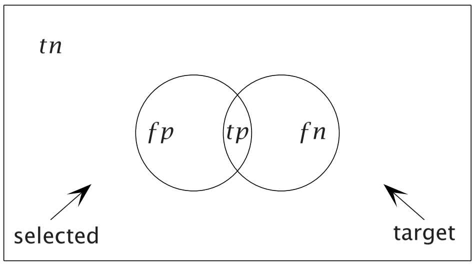
where P is precision, R is recall and α weights precision and recall. (Or in terms of β, where $\alpha = 1/(\beta^2 + 1)$.)

A value of $\alpha = 0.5$ is often chosen.

$$F = \frac{2PR}{R + P}$$

At break-even point, when $R = P$, then $F = R = P$

A diagram motivating the measures of precision and recall.



Accuracy is not a useful measure when the target set is a tiny fraction of the total set.

A combined measure

Does one just average them? Bad, because the measures aren't independent.

What's a sensible model?

Rijsbergen (1979:174) defines and justifies the usually used alternative.

(see http://www.dcs.gla.ac.uk/Keith/Preface.html)

Ways of averaging

Precision	Recall	Arithmetic	Geometric	Harmonic	Minimum
80	10	45	28.3	17.8	10
80	20	50	40.0	32.0	20
80	30	55	49.0	43.6	30
80	40	60	56.6	53.3	40
80	50	65	63.2	61.5	50
80	60	70	69.3	68.6	60
80	70	75	74.8	74.7	70
80	80	80	80.0	80.0	80
80	90	85	84.9	84.7	80
80	100	90	89.4	88.9	80

The Vector Space Model (Salton, TREC)

Represents terms and documents as vectors in k-dimen. space based on the bag of words they contain:

d = The man said that a space age man appeared

d' = Those men appeared to say their age

$$\vec{d} = \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{pmatrix}$$

	$\vec{d}$	$\vec{d'}$
age	1	1
appeared	1	1
man	2	0
men	0	1
said	1	0
say	0	1
space	1	0

Text categorization/classification

- Take a document and assign it a label representing its topic
- Have some predefined categories for texts
  - Classic example: decide if a newspaper article is about politics, business, or sports
  - Predefined categories for news items on newswires – Reuters categories
  - ACM keywords
  - Yahoo! or Open Directory classes (extra complexity: hierarchical)
  - UMLS (Unified Medical Language System) MeSH codes
- Generally approached using a bag of words model

Text categorization/classification

- There are many other uses for the same technology:
  - Is this page a laser printer product page?
  - Does this company accept overseas orders?
  - What kind of job does this job posting describe?
  - What kind of position does this list of responsibilities describe?
  - What position does this this list of skills best fit?
  - Word sense disambiguation can actually be thought of as text (here, context) categorization:
    - Is this the “computer” or “harbor” sense of port?

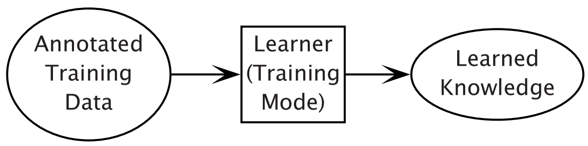
Text categorization/classification

- This can be thought of as working out the “meaning” of documents in a very primitive sense: for instance, a news article “means” *Europe* and *politics*
- Most of the companies that claim to be understanding your document, doing eCRM processing of your users complaining emails, etc., are essentially offering text categorization at the moment:
  - Kana
  - Autonomy
  - Bhasha
  - Banter

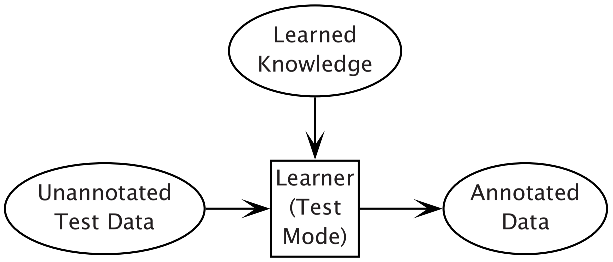
NLP as a classification problem

- Central to recent advances in NLP has been reconceptualizing NLP as a machine learning classification problem
- We – preferably someone else – hand-annotate data, and then learn using standard ML methods
- Annotated data items are feature vectors $\vec{x}_i$ with a classification c_i.
- Our job is to assign an unannotated data item $\vec{x}$ to one of the classes c_k (or possibly to the doubt $\mathcal{D}$ or outlier $\mathcal{O}$ categories – though in practice rarely used).

Training procedure



Testing = deployed procedure

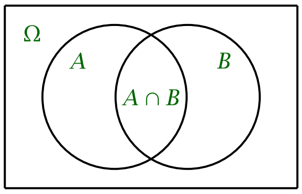


Statistical decision theory

- The proportion of items in each class c_k has a prior probability π_k.
- Feature vectors $\vec{x}$ have distribution $P_k(\vec{x})$ for class c_k
- Assume (falsely) that these are known
- **Bayes optimal classifier / Bayes (decision) rule**:
  - Simple version: For an item $\vec{x}$, assign class $\boldsymbol{c = \arg\max_{c_k} P(c_k|\vec{x})}$
  - More complex versions put in explicit loss functions for making different kinds of mistakes, and decide $\mathcal{D}$ if no class is likely.

Bayesian classification in WSD

- Vectors $\vec{x}$ are some representation of the context of use of a word
- Problem: We usually don't know $P(c_k|\vec{x})$
- Partial solution: we break it up by Bayes' rule



$$P(B|A) = P(A \cap B)/P(A)$$

$$P(A \cap B) = P(B)P(A|B)$$

Bayes' theorem

$$P(B|A) = \frac{P(A,B)}{P(A)} = \frac{P(A|B)P(B)}{P(A)}$$

If $A \subseteq \cup_i B_i$ and the B_i are disjoint, then (sum rule):

$$P(A) = \sum_i P(A,B_i) = \sum_i P(A|B_i)P(B_i)$$

Bayes' theorem: If $A \subseteq \cup_{i=1}^n B_i$, $P(A) > 0$, and $B_i \cap B_j = \emptyset$ for $i \neq j$:

$$\underbrace{P(B_j|A)}_{\text{Posterior}} = \frac{P(A|B_j)P(B_j)}{P(A)} = \frac{\overset{\text{Likelihood}}{P(A|B_j)} \times \overset{\text{Prior}}{P(B_j)}}{\underbrace{\sum_{i=1}^n P(A|B_i)P(B_i)}_{\text{Normalizing term}}}$$

Naive Bayes Text Categorization

- $\vec{x}$ is our document, a list of words
- c_k is the hidden class

$$\begin{aligned} c' &= \arg\max_{c_k} P(c_k|\vec{x}) \\ &= \arg\max_{c_k} \frac{P(\vec{x}|c_k)}{P(\vec{x})} P(c_k) \\ &= \arg\max_{c_k} P(\vec{x}|c_k)P(c_k) \\ &= \arg\max_{c_k} [\log P(\vec{x}|c_k) + \log P(c_k)] \end{aligned}$$

Naive Bayes models

- The *Naive Bayes assumption* is that the attributes used for description are all conditionally independent:

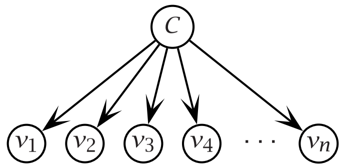
  Naive Bayes assumption

  $P(\vec{x}|c_k) = P(\{v_j | v_j \text{ in } \vec{x}\}|c_k) = \prod_{v_j \text{ in } \vec{x}} P(v_j|c_k)$
- This is commonly referred to as the *bag of words* assumption
- **Decision rule for Naive Bayes**

  Decide c' if $c' = \arg\max_{c_k} [\log P(c_k) + \sum_{v_j \text{ in } \vec{x}} \log P(v_j|c_k)]$
- Note that there are two Naive Bayes models (McCallum and Nigam 1998)

Two Naive Bayes models: Multinomial

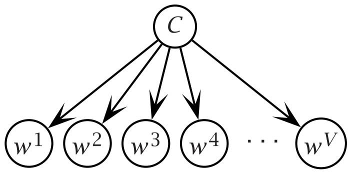
- v_j is word j of the context



- Model of Gale et al. (1992) (for WSD). Usual in StatNLP.
- The CPT for each multinomial is identical (tied parameters)
- The multinomial is estimated over the whole vocabulary.

Two Naive Bayes models: Bernoulli

- w^j is word (type) j of the vocabulary of features



- Each feature is binary yes/no (though could be count/range)
- Model normally presented in the graphical models literature
- Generally (but not always) performs worse
- Requires careful and aggressive feature selection

Naive Bayes models

- Feature selection: commonly χ^2 or mutual information, but there are better methods to find non-overlapping features (Koller and Sahami 1996). Only important in Bernoulli model.
- Naive Bayes is simple, but often about as good as there is (Domingos and Pazzani 1997)
- There are successful more complex probabilistic classifiers, particularly TAN – Tree Augmented Naive Bayes (Friedman and Goldszmidt 1996)
- One can get value from varying context size according to type of word being disambiguated (commonly: noun is big context, verb is small context)

Text classification results

- You've got to be careful that you're comparing apples and apples
- One source of careful recent comparative studies is Yang (1999a *Information Retrieval*, 1999b *SIGIR*)
- Results on Reuters for which several versions have been prepared commonly differed by as great a percentage between *versions* as between *methods*.
- Many comparisons were invalidated by data differences.
- (Should do statistical significance tests, confidence intervals, ...)

Averaging

- Two ways of computing an average give very different results!
- *Micro-averaging* make a single contingency table for all the data by summing the scores in each cell for all categories, and evaluate it
  - Gives equal weight to each object in the collection
  - Score is dominated by large categories
- *Macro-averaging:* compute measure for each category contingency table separately and then average the evaluation measure over categories
  - Gives equal weight to each category
  - Gives a better sense of the quality of classification across all categories.

Yang SIGIR results

- Micro-averaging (dominated by performance on large categories):
  - $SVM > kNN \gg \{LLSF, NNet\} \gg NB$
  - F: 0.860, 0.857, 0.850, 0.829, 0.80
- Macro-averaging
  - $\{SVM, kNN, LLSF\} \gg \{NNet, NB\}$
  - F: 0.525, 0.524, 0.501, 0.377, 0.389
- Yang doesn't test MaxEnt (loglinear) models, but other results (Nigam et al. 1999) show it doing much better than NB, so it appears to belong in the top group.