Evaluation, Annotation and Information Retrieval

Kan Min-Yen

Day 2 / Morning

Review

- The central NLP issue = Ambiguity
- Natural language is not context free

- Heuristic methods for speed and where limited flexibility is needed
- Machine learning methods for robustness against noise
  - –Needing clean training data

Day Outline

Day 1	>> Day 2	Day 3
AM	**AM**	**AM**
– Applications' Input / Output	– Evaluation	– Sequence Labeling
– Resources	– Annotation	– CRF++ Hands-on
	– Information Retrieval	
	– ML Intro	
PM	**PM**	**PM**
– Selected Toolkits	– Machine Learning	– Dimensionality Reduction
– Python Intro	– SVM Hands-on	– Clustering
– NLTK Hands-on		– Trends & Issues

Day Outline

- Evaluation
- Annotation
- Information Retrieval
- NLP as Machine Learning: crash course on ML

- Rule-based vs. statistical NLP
- Statistical Modeling Paradigms
- Hands-on: Text Classification with SVMLight

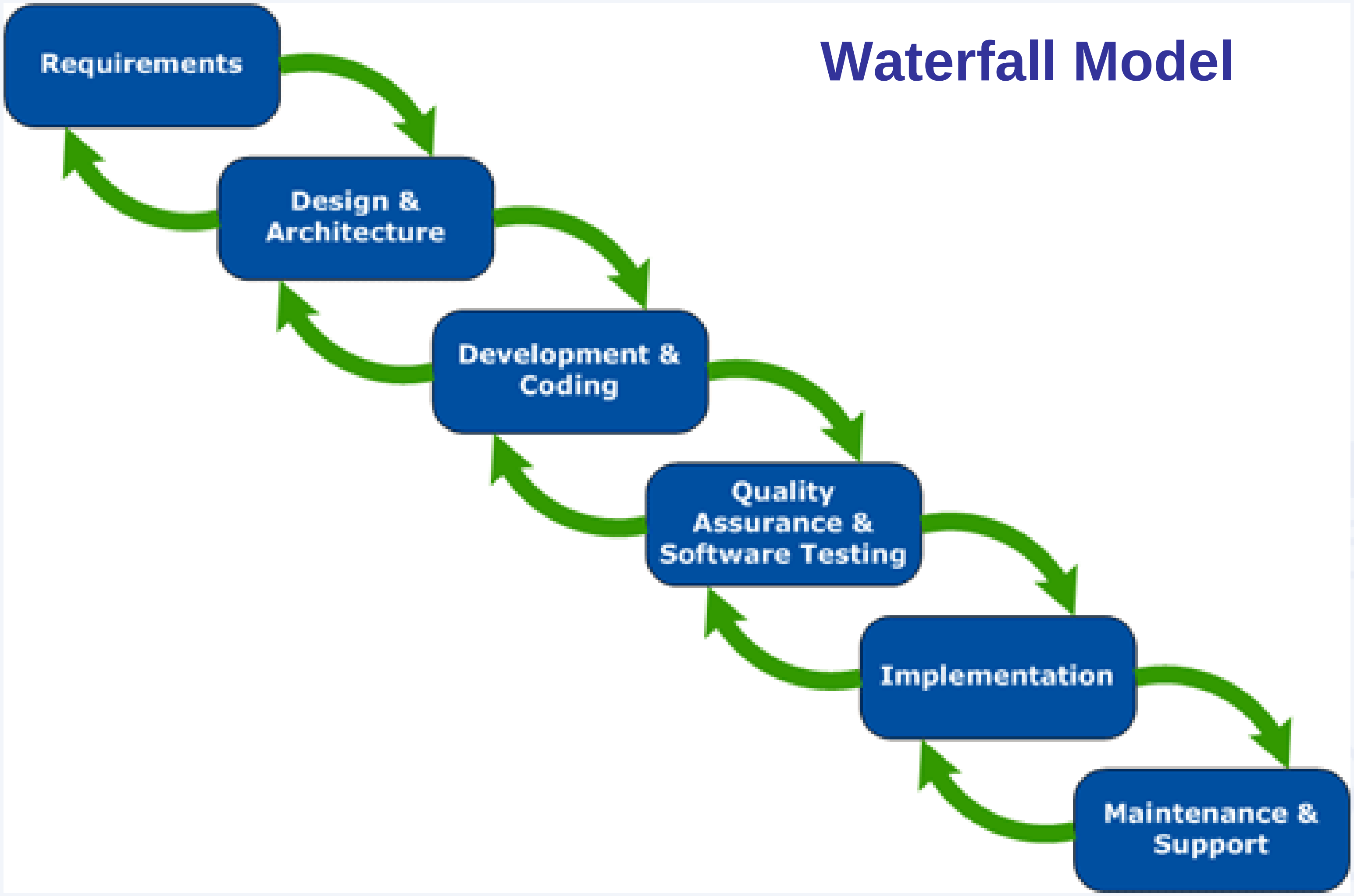
Evaluation

Design models

With ground truth

Without ground truth?

Application development cycle



Parallels in NLP application design

- **Requirements**
- **Design and Architecture**
- **Development and Coding**
- **Quality, Assurance and Software Testing**
- **Implementation**
- **Maintenance and Support**

- **Obtain an annotated corpus**
- **Build a baseline model**
- **Repeat:**
  - Analyze the most common errors
  - Find out what information could be helpful
  - Modify the model to exploit this information:
    - Use new features
    - Change the model

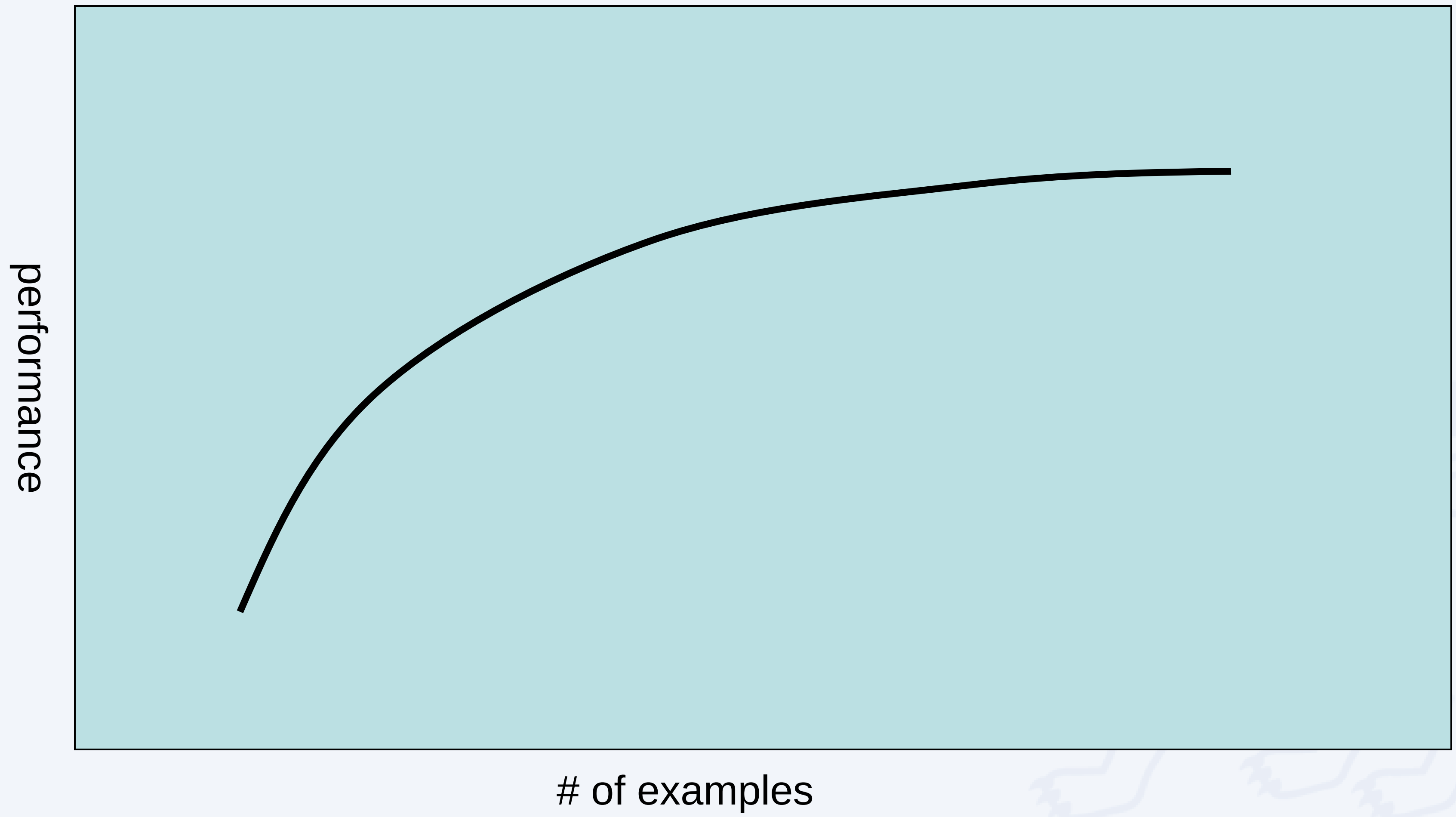
Evaluation phenomenon

1. **Cascading Errors**
   - Stringing together modules creates larger downstream errors
   - Most accuracy needed upstream, preprocessing tasks have large effect on output

     Don't neglect preprocessing, may change architecture downstream

2. **A little (clean) data goes a long way**
   - Corollary: More data has less of an effect
   - Must be from the same distribution / target domain
   - Assess performance trend over subsets of data

Improvement with respect to data size



Evaluation Types

Intrinsic

- Assess directly on ground truth

Extrinsic

- Assess against other tasks
- Useful to determine suitability for application
- Subjectivity factors
- Examples:
  - change in revenue via marketing
  - summaries via question answering

Evaluation Contingency Table

	System says is relevant	System says is irrelevant
Document is actually relevant	**TP** (True Positive)	**FN** (False Negative)
Document is actually irrelevant	**FP** (False Positive)	**TN** (True Negative)

Evaluation Metrics

$$\frac{TP}{TP+FP}$$

- **Precision = Positive Predictive Value**
  - "ratio of the number of relevant documents retrieved over the total number of documents retrieved"
  - how much extra stuff did you get?

$$\frac{TP}{TP+FN}$$

- **Recall = Sensitivity**
  - "ratio of relevant documents retrieved for a given query over the number of relevant documents for that query in the database"
  - how much did you miss?

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

- **F1 measure = harmonic mean of P and R**
  - Question: Why harmonic average?
  - Can use other coefficients instead of 1

One number to rule them all: MAP

- **A "standard" measure: Mean Average Precision (MAP)**
  - Average of precision at all points where a new relevant document is found.
  - Problem: favors systems with high recall
  - On the web, a user is usually looking just at the first a few results in Web search.

    Leads to precision at *k* documents, but it's kludgy: not sensitive to the ranking of every relevant document.

A second try: nDCG

- "Gain": **Each rel doc gives some level of relevance to the user**
  G' = <3,2,3,0,0,1>

- "Cumulative": **overall utility of *n* docs = sum of gain of each rel doc.**
  CG' = <3,5,8,8,8,9>

- "Discount" **docs further down in list, as they are less likely to be used**
  DCG' = <3, 3+2/log2, 3+2/log2+3/log3, …, 3+2/log2+3/log3+1/log6>

- "Normalized" **against ideal IR system rankings**
  Ideal G' = <3,3,2,1,0,0>
  Ideal DCG' = <3, 3+3/log2, 3+3/log2+2/log3, 3+3/log2+2/log3+1/log4, …>
  nDCG' = DCG' / Ideal DCG' = <1, …>

Pro: works naturally from fractional relevance
Con: have to set the discounting coefficients in NDCG (why log?)

What constitutes good clustering?

- **Internal criterion: A good clustering will produce high quality clusters in which:**
  - –the intra-class (that is, intra-cluster) similarity is high
  - –the inter-class similarity is low
  - –The measured quality of a clustering depends on both the document representation and the similarity measure used

NUS National University of Singapore

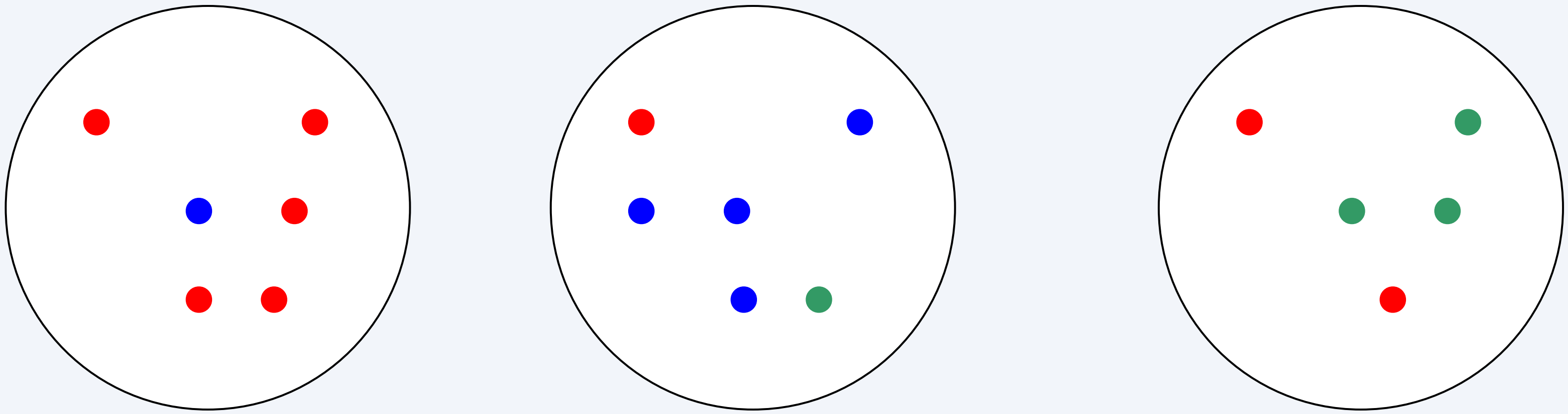
Cluster Quality Evaluation

- Simple measure: purity, the ratio between the dominant class in the cluster π_i and the size of cluster ω_i

$$Purity(\omega_i) = \frac{1}{n_i} \max_j (n_{ij}) \quad j \in C$$

- Others are entropy of classes in clusters (or mutual information between classes and clusters)

Purity example



Cluster I Cluster II Cluster III

Cluster I: Purity = 1/6 (max(5, 1, 0)) = 5/6

Cluster II: Purity = 1/6 (max(1, 4, 1)) = 4/6

Cluster III: Purity = 1/5 (max(2, 0, 3)) = 3/5

Segmentation

G: Xxx xxx xxx xxx xxx, President Bush yyy yyy

S1: Xxx xxx xxx xxx xxx, President Bush yyy yyy

S2: Xxx xxx xxx xxx xxx, President Bush yyy yyy

S1 and S2 equally bad by exact P/R calculations

But surely S2 better than S1 (not off by so much)

P_k and WindowDiff (and other metrics) account for this

- WindowDiff implemented in NLTK

N-gram metrics: BLEU, ROUGE

- For machine translation (BLEU), summarization (ROUGE)
- Against a reference translation
- Metrics count the number of overlapping units

ROUGE-N: N-gram co-occurrence statistics is a recall oriented metric

G- police killed the gunman

S1- police kill the gunman

S2- the gunman kill police

S1 equivalent to S2

ROUGE-L: Based on longest common subsequence

G - police killed the gunman

S2- police kill the gunman

S3- the gunman kill police

S1 better than S2

Evaluation without ground truth

Possible?

- **Not really, not able to judge:**
  - Level of performance or worse
  - How to improve

- **In practice, can't afford much annotation effort**
- **Alternatives?**

Analysis: Macro vs. Micro

Macro: Summative

- **First stage of analysis**
- **Assess over all data**
- **Impressionistic**

- **Useful for presenting %ages**
- **Useful in identifying areas for microanalysis using contingency table**
- **NOT useful in improving data**

Micro: Formative

- **Look at specific examples**
  - Sample accordingly (randomize)
- **Categorize errors J&M pp 313**
  - Hard work to come up with error categories (*cf* annotation)
  - Upstream errors?
- **Fix**
  - Create within model
  - Pre or post-process

Ideas from (machine) learning

- **Bootstrapping**
  - Label a little, assume most confident automatically classified data is correct, retrain
- **Co-training**
  - Train two different models (with different features)
  - Have them learn from each other
- **Active Learning**
  - Take least confident auto classified data and manually annotate it

Improving NLP applications' accuracy

- **Obtain an <1: annotated corpus>**
- **Build a baseline model**
- **Repeat:**
  - – Analyze the most common errors
  - – Find out what information could be helpful
  - – Modify the model to exploit this information:

    Use **<2: new features>**

    Change the **<3: model>**

- **Increase the data**
  - – Size of examples covered
  - – Cleanliness of annotation
  - – Representiveness
- **Features**
  - – Microanalysis
  - – Decision Tree / Regression
  - – Code new features using SWOT
- **Model**
  - – Try a different model

Annotation

9 guidelines to follow

Annotation Toolkits

Annotation Guidelines

1. **State policy**
   - In addition to a technical one (e.g., DTD)
   - lumping vs. splitting
   - Use of kitchen sink (“other”) category
   - Be clear about what the data is / will be used for

2. **State guidelines clearly**
   - Embedded or standoff annotation?
   - What are elements? What are attributes?
     e.g., POS tags versus chunking
   - Ensure coding dimensions
   - Crossing constraints? Discontinuous elements?
   - Aids training new personnel later

Annotation Guidelines

3. **Define terminology in a glossary**
   - Then use it consistently

4. **Show difficult and confusable examples with cross-references**
   - Example: Penn Discourse Treebank Guidelines

5. **Use an odd number of subjects**

6. **Strive for consistency**
   - Better to have fewer annotators work more

Annotation Guidelines

7. **Assign same number of subjects to each data point for validation**
   - Once it's clear that agreement level on par, create separate data segment for annotation per subject

8. **Randomized but stable order for annotation**

9. **Be prepared for annotators to fail**
   - Annotators do drop out; will have to train new ones
   - Keep records of who annotated what

Computing Agreement

- **Several methods**
  - Kendall's Tau
  - Pearson Correlation

- **Analyze your confusion matrix**
  - Helps you to correct and strengthen your guidelines

Annotation Formats

- SGML
- XML
- TEI and TEI-lite
- JSON / YAML
- Domain specific markup

Annotation Toolkits

http://www.ldc.upenn.edu/annotation/

http://www.exmaralda.org/annotation

- **Alembic (MITRE)**
- **ACE Annotation toolkit (LDC)**
  - Relation Tagging / IE
- **Atlas TI**
  - Commercial ($$) toolkit used to annotate multimedia

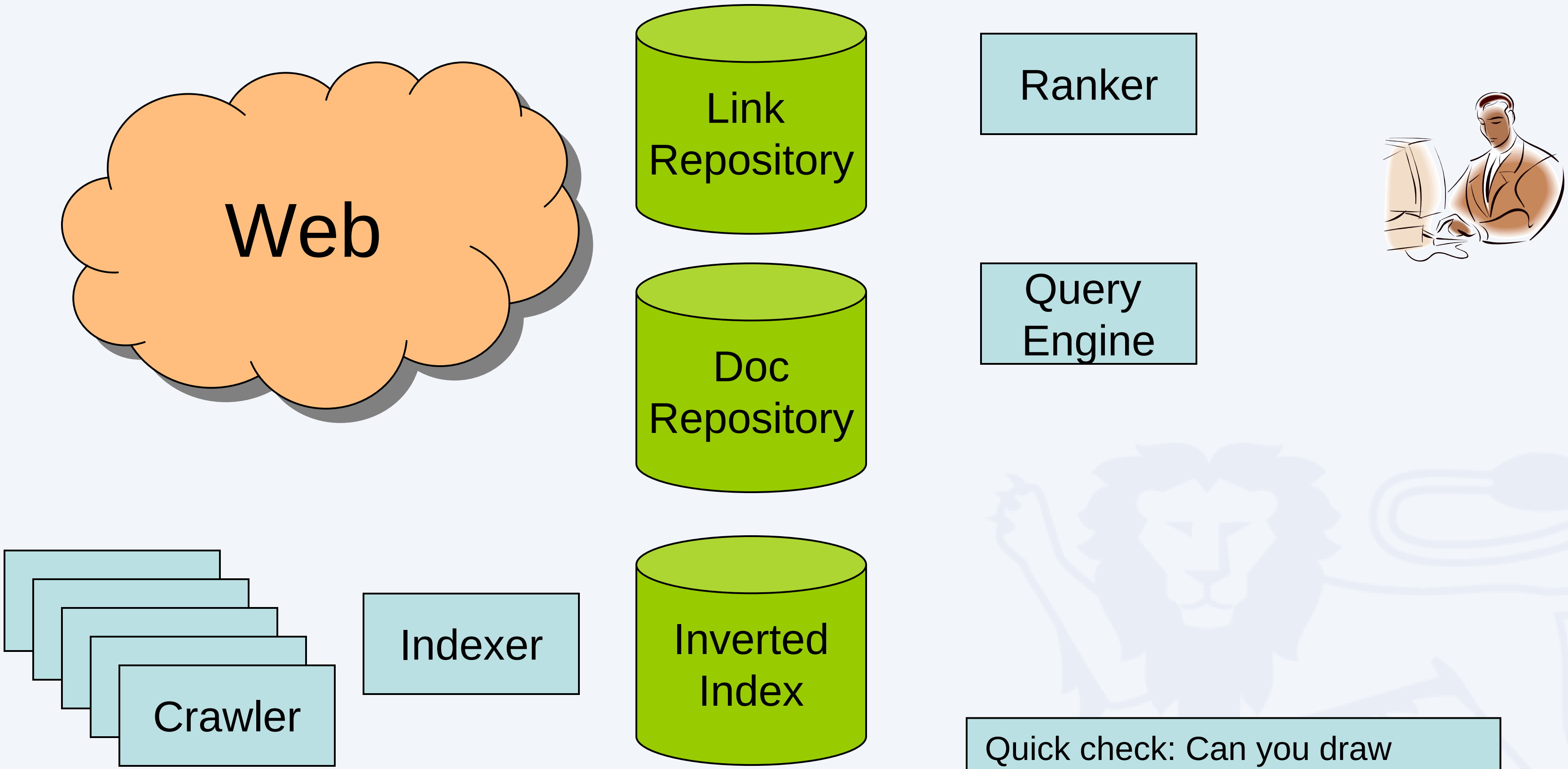
Information Retrieval

Search Engines

Word Weighting

Documents and queries as vectors

The anatomy of a search engine



Quick check: Can you draw the links and label them yourself?

Doc Representation

Sad but true

Query and documents seen as a bag of words

Matching is done by comparing these BoWs

How do we get to a BoW given a text?

Let's look at unstructured text first:

- **Tokenization - not all languages have spaces to delimit**
  - what about phrases like GermanNounCompounds
  - HTML structure can help to recover latent semi structure but is not guaranteed to be well formed

Doc Representation

- **Stemming - recover stem for agglutinative languages**
  - –For English: Porter and Lovins stemmer: uses 5 iterations to strip suffixes. Does not necessarily result in a word
  - –What's a "stem" in CJK?
- **Case Folding - combine the same word in different cases: next NEXT Next NeXT**
- **Stop Words - remove frequent words that are not used in queries.**

Which of 2 of these three attack the same problem?
What is this problem?

Term Specific Weighting

Xxxxxxxxxxxxxx IBM xxxxxxxxxxx xxxxxxxx xxxxxxxxxxx IBM xxxxxxx xxxxxxxxxx xxxxxxxx Apple. Xxxxxxxxxx xxxxxxxxxx IBM xxxxxxxx. Xxxxxxxxxx xxxxxxxx Compaq. Xxxxxxxxx xxxxxxx IBM.

- **We call this Term Frequency**

  although this is really just a count

- **Forms of TF_{ij} =**

  N_{ij}

  $1+\ln(N_{ij})$

  $N_{ij}/\max(N_i)$

Document Specific Weighting

- **Which of these tells you more about a doc?**
  - –10 occurrences of *hernia*?
  - –10 occurrences of *the*?
- **Would like to attenuate the weight of a common term**
  - –But what is “common”?
- **Suggest looking at collection frequency (*cf*)**
  - –The total number of occurrences of the term in the entire collection of documents

Document frequency

- **But document frequency (*df*) may be better:**
- ***df* = number of docs in the corpus containing the term**

Word	***cf***	***df***
ferrari	**10422**	**17**
insurance	**10440**	**3997**

- **Document/collection frequency weighting is only possible in known (static) collection.**
- **So how do we make use of *df* ?**

This is tf.idf

- **tf.idf measure combines:**
  - –term frequency (*tf*)
    - or *wf*, some measure of term density in a doc
  - –inverse document frequency (*idf*)
    - measure of informativeness of a term: its rarity across the whole corpus
    - could just be raw count of number of documents the term occurs in ($idf_i = 1/df_i$)
    - but by far the most commonly used version is:

$$idf_i = \log\left(\frac{n}{df_i}\right)$$

- **Justified as optimal weight w.r.t relative entropy**

Documents as vectors

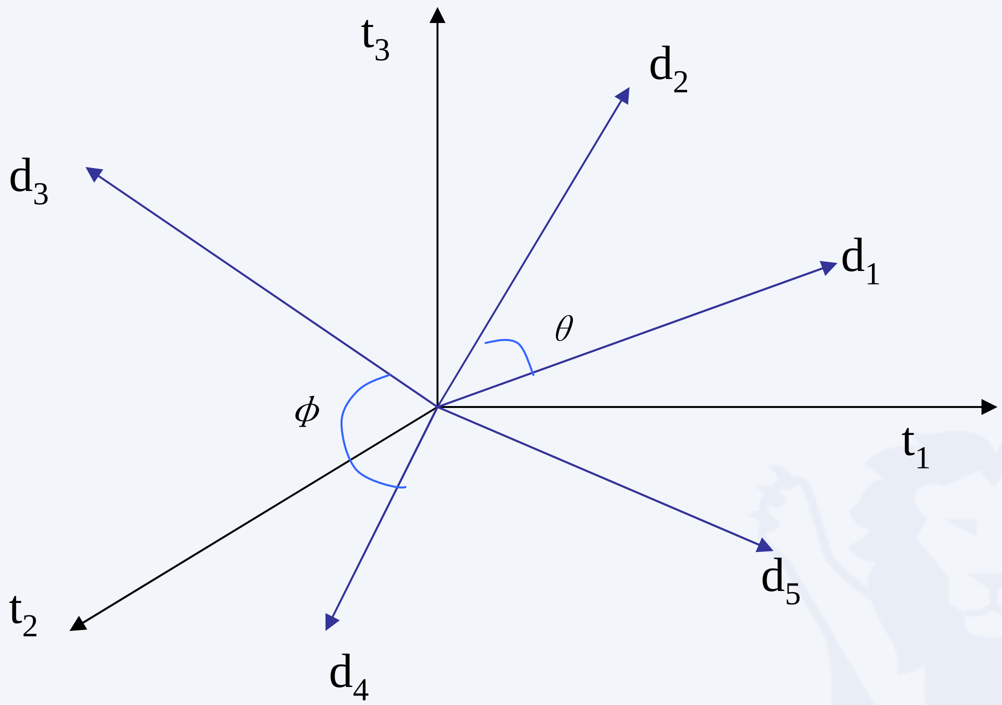
- **Each doc *j* can now be viewed as a vector of *tf x idf* values, one component for each term**

- **So we have a vector space**
  - terms are axes
  - docs live in this space
  - even with stemming, may have 20,000+ dimensions

Why turn docs into vectors?

- **First application: Query-by-example**
  - –Given a doc *d*, find others “like” it.
- **Now that *d* is a vector, find vectors (docs) “near” it.**

Intuition



Postulate: Documents that are “close together” in the vector space talk about the same things.

Desiderata for proximity

- If d_1 is near d_2, then d_2 is near d_1.
- If d_1 near d_2, and d_2 near d_3, then d_1 is not far from d_3.
- No doc is closer to d than d itself.

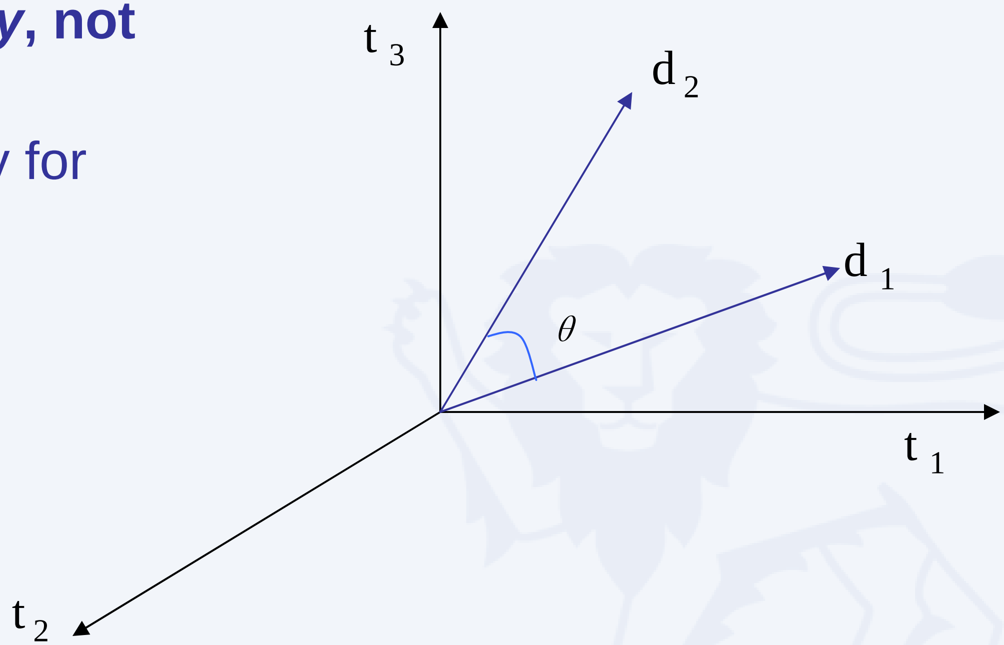
First cut

- **Idea: Distance between d_1 and d_2 is the length of the vector $|d_1 - d_2|$.**
  - –Euclidean distance
- **Why is this not a great idea?**

- **We still haven't dealt with the issue of length normalization**
  - –Short documents would be more similar to each other by virtue of length, not topic
- **However, we can implicitly normalize by looking at *angles* instead**

Cosine similarity

- **Distance between vectors d_1 and d_2 *captured* by the cosine of the angle x between them.**
- **Note – this is *similarity*, not distance**
  - No triangle inequality for similarity.



Cosine similarity

A vector can be *normalized* (given a length of 1) by dividing each of its components by its length – here we use the *L_2* norm

$$\|\mathbf{x}\|_2 = \sqrt{\sum_i x_i^2}$$

This maps vectors onto the unit sphere:

Then,

$$\left|\vec{d}_j\right| = \sqrt{\sum_{i=1}^{n} w_{i,j}} = 1$$

Longer documents don't get more weight

Cosine similarity

$$sim(d_j, d_k) = \frac{\vec{d}_j \cdot \vec{d}_k}{\left|\vec{d}_j\right| \left|\vec{d}_k\right|} = \frac{\sum_{i=1}^{n} w_{i,j} w_{i,k}}{\sqrt{\sum_{i=1}^{n} w_{i,j}^2} \sqrt{\sum_{i=1}^{n} w_{i,k}^2}}$$

- **Cosine of angle between two vectors**
- **The denominator involves the lengths of the vectors.**

Normalization

Normalized vectors

- For normalized vectors, the cosine is simply the dot product:

$$\cos(\vec{d}_j, \vec{d}_k) = \vec{d}_j \cdot \vec{d}_k$$

Example

- **Docs: Austen's *Sense and Sensibility*, *Pride and Prejudice*; Bronte's *Wuthering Heights. tf* weights**

	SaS	PaP	WH
affection	115	58	20
jealous	10	7	11
gossip	2	0	6

	SaS	PaP	WH
affection	0.996	0.993	0.847
jealous	0.087	0.120	0.466
gossip	0.017	0.000	0.254

- cos(SAS, PAP) = .996 x .993 + .087 x .120 + .017 x 0.0 = 0.999
- cos(SAS, WH) = .996 x .847 + .087 x .466 + .017 x .254 = 0.889

Cosine similarity exercise

- ***Exercise: Rank the following by decreasing cosine similarity. Assume tf.idf weighting:***
  - –Two docs that have only frequent words ***(the, a, an, of)*** in common.
  - –Two docs that have no words in common.
  - –Two docs that have many rare words in common ***(wingspan, tailfin).***

Phrase queries

- **Running multiple queries**
  - Backoff to n-1 gram in case of too few results

    "A B C"

    "A B", "B C"

    A, B, C

- **Proximity as window *w* between term occurrences**
  - Prefer the window to be smaller

Relevance Feedback

- **Main Idea:**
  - Modify existing query based on relevance judgements
    - Extract terms from relevant documents and add them to the query
    - and/or re-weight the terms already in the query
  - Two main approaches:
    - Automatic (pseudo-relevance feedback)
    - Users select relevant documents
  - Users/system select terms from an automatically-generated list

Relevance Feedback

- **Usually do both:**
  - Expand query with new terms
  - Re-weight terms in query

- **There are many variations**
  - Usually **positive weights** for terms from **relevant** docs
  - Sometimes **negative weights** for terms from **non-relevant** docs
  - Select terms sometimes by requiring them to match query in addition to document

Rocchio Method

$$Q_1 = Q_0 + \beta \sum_{i=1}^{n_1} \frac{R_i}{n_1} - \gamma \sum_{i=1}^{n_2} \frac{S_i}{n_2}$$

where

$Q_0 =$ the vector for the initial query

$R_i =$ the vector for the relevant document i

$S_i =$ the vector for the non-relevant document i

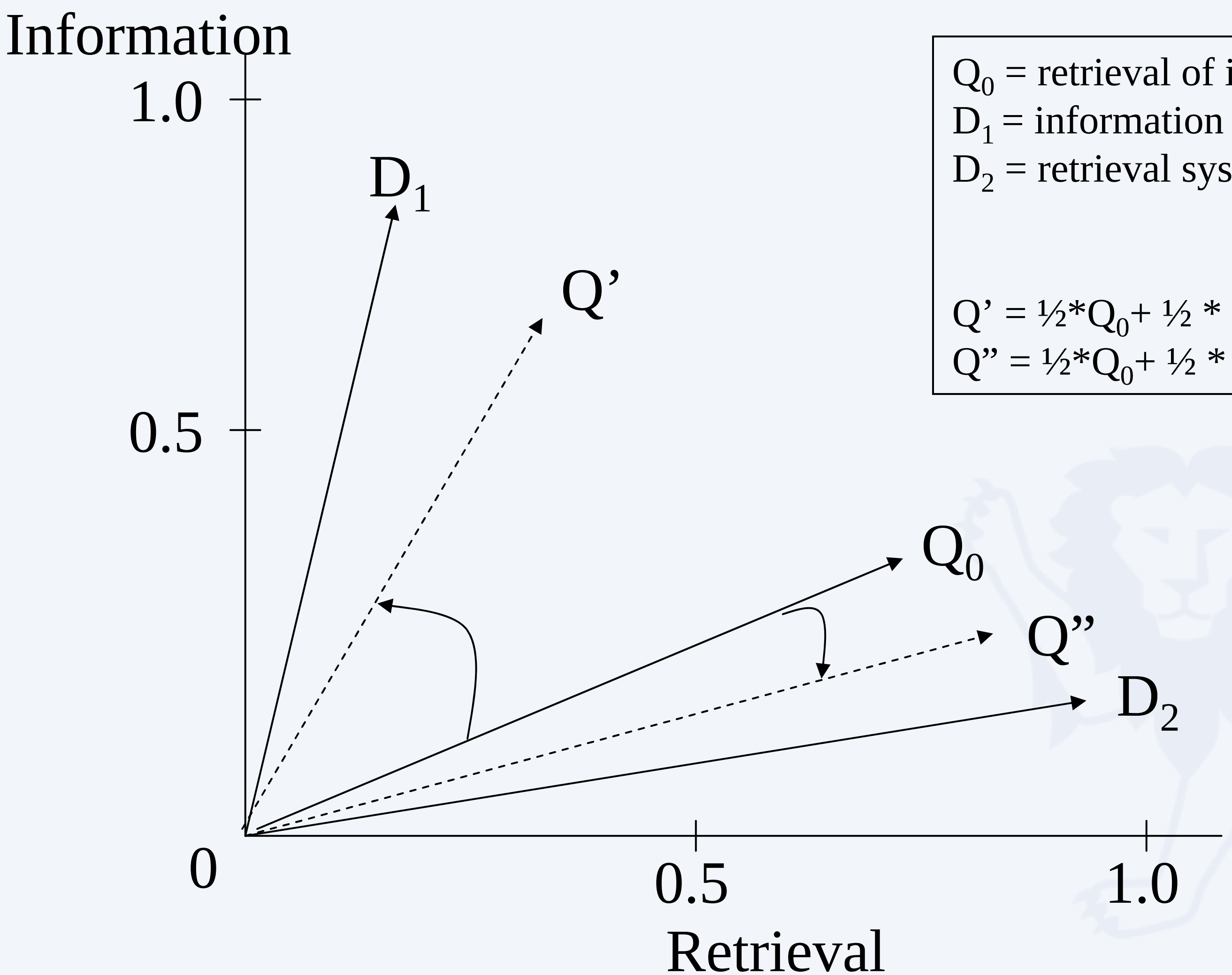
$n_1 =$ the number of relevant documents chosen

$n_2 =$ the number of non-relevant documents chosen

β and γ tune the importance of relevant and nonrelevant terms

(in some studies best to set β to 0.75 and γ to 0.25)

Rocchio/Vector Illustration



Q_0 = retrieval of information = (0.7,0.3)
D_1 = information science = (0.2,0.8)
D_2 = retrieval systems = (0.9,0.1)

Q' = ½*Q_0+ ½ * D_1 = (0.45,0.55)
Q" = ½*Q_0+ ½ * D_2 = (0.80,0.20)

Concepts from Machine Learning

Inductive learning

Simplest form: learn a function from examples

- ***f*** **is the target function**
- **An example is a pair (*x*, *f(x)*)**

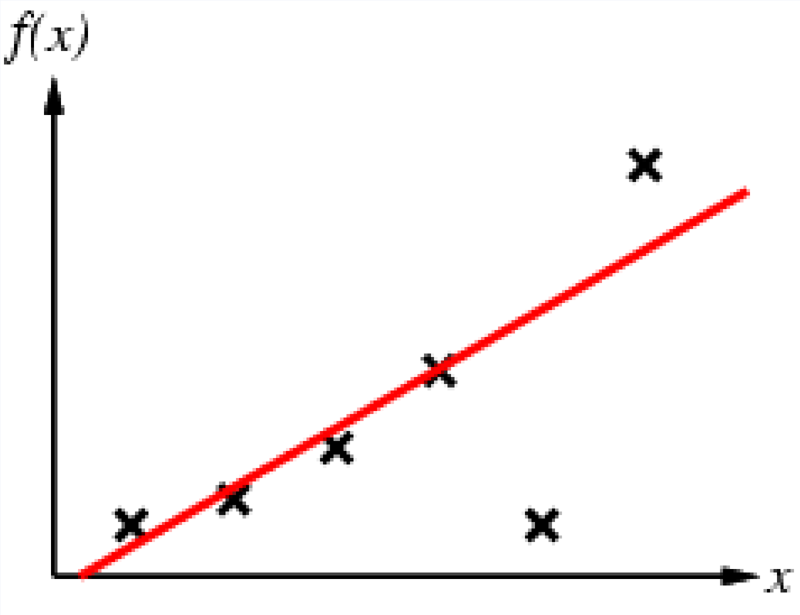
- **Problem: find a hypothesis *h***
  such that $h \approx f$
  given a training set of examples

- **Many learners do this by constructing a generalized representation of the training set called a model**

Inductive learning method

- Construct/adjust *h* to agree with *f* on training set
- (*h* is consistent if it agrees with *f* on all examples)

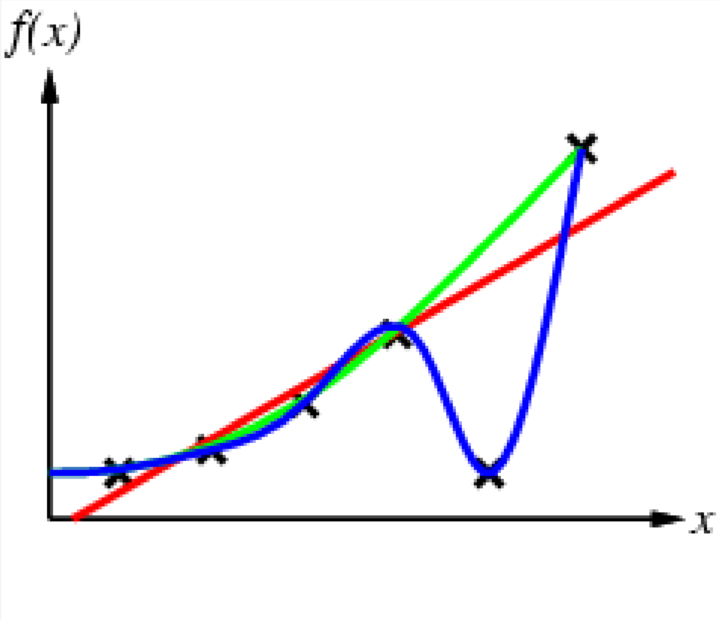
- E.g., curve fitting:



Inductive learning method

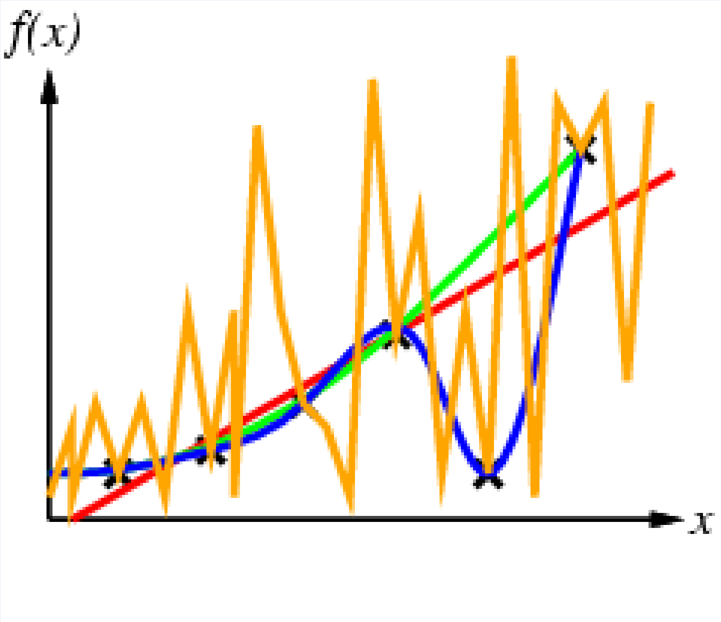
- Construct/adjust *h* to agree with *f* on training set
- (*h* is consistent if it agrees with *f* on all examples)

- E.g., curve fitting:



Inductive learning method

- What’s to stop us from predicting this?



- Ockham’s razor: prefer the simplest hypothesis consistent with data

Turn your task into a learning problem

Many tasks can be transformed into a learning problem

- **Transform the data into features**
- **Represent the outcomes as a classification task**

Overview of learning

- **Learners deal with multiple pieces of evidence**
  - –x can be a vector of values instead of a single value
  - –These vectors can be very large
  - –Length of the vector = dimensionality
- **Learners deal with numeric data**
  - –Textual data has to be transformed into numeric features
  - –Each text token can be reflected as a separate vector
- **Learners deal with a fixed set of classes**
  - –(e.g., f(x) = {finance, politics, sports}
  - –But some do this by decomposing multiple classes into n way binary problems, not always optimal

Procedure

Annotation (tedious part)

- –Determine data set and classification
- –Label the data with the correct classifications
  - This can sometimes be done semi-automatically

Coding (thinking part)

- –Code features related to the classification
- –Choose an appropriate learning algorithm

Test time

- –Split datasets into training and testing portions
- –Determine training and testing error
- –Analyze errors

Training and testing sets

- **Where does the test set come from?**
  1. Collect a large set of examples
  2. Divide into **training** and **testing data**
  3. Train on training data, assess on testing
  4. Repeat 1-3 for different splits of the set.

  The above is called **cross-validation**.

- **Must be from the same distribution!!**

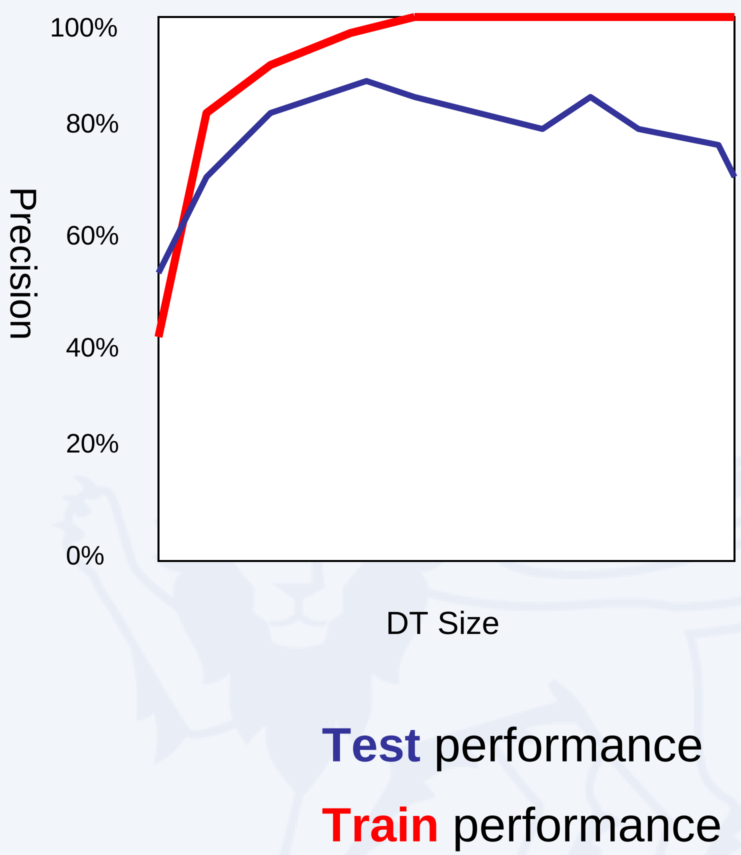
  "Learning … enable[s] the system to do the task or tasks drawn from the same population**" – Herb Simon**

  – To think about: Why?

  – Related area: domain adaptation

Overfitting

- **Better training performance = test performance?**
- **Nope. Why?**
  1. Hypothesis too specific
  2. Models noise
- **Pruning**
  – Keep complexity of hypothesis low
  – Stop splitting when:
    IC below a threshold
    Too few data points in node



Summary

- **Evaluation**
  - Subjectivity and ambiguity problems in evaluation
  - Transfer evaluation into an objective measure
    - Measure whether objective metric improvement correlates with subjective improvement
- **Annotation**
  - Largely XML-centric
  - Follow best practices to get the most bang for the bank
- **Information Retrieval**
  - Weighting words to take local, global importance into account
  - Define docs and queries as vectors to compute similarity
  - Much more here: weighting using hyperlinks
- **Machine Learning**
  - Learning a function on inputs to outputs
  - Prefer the simplest consistent hypothesis