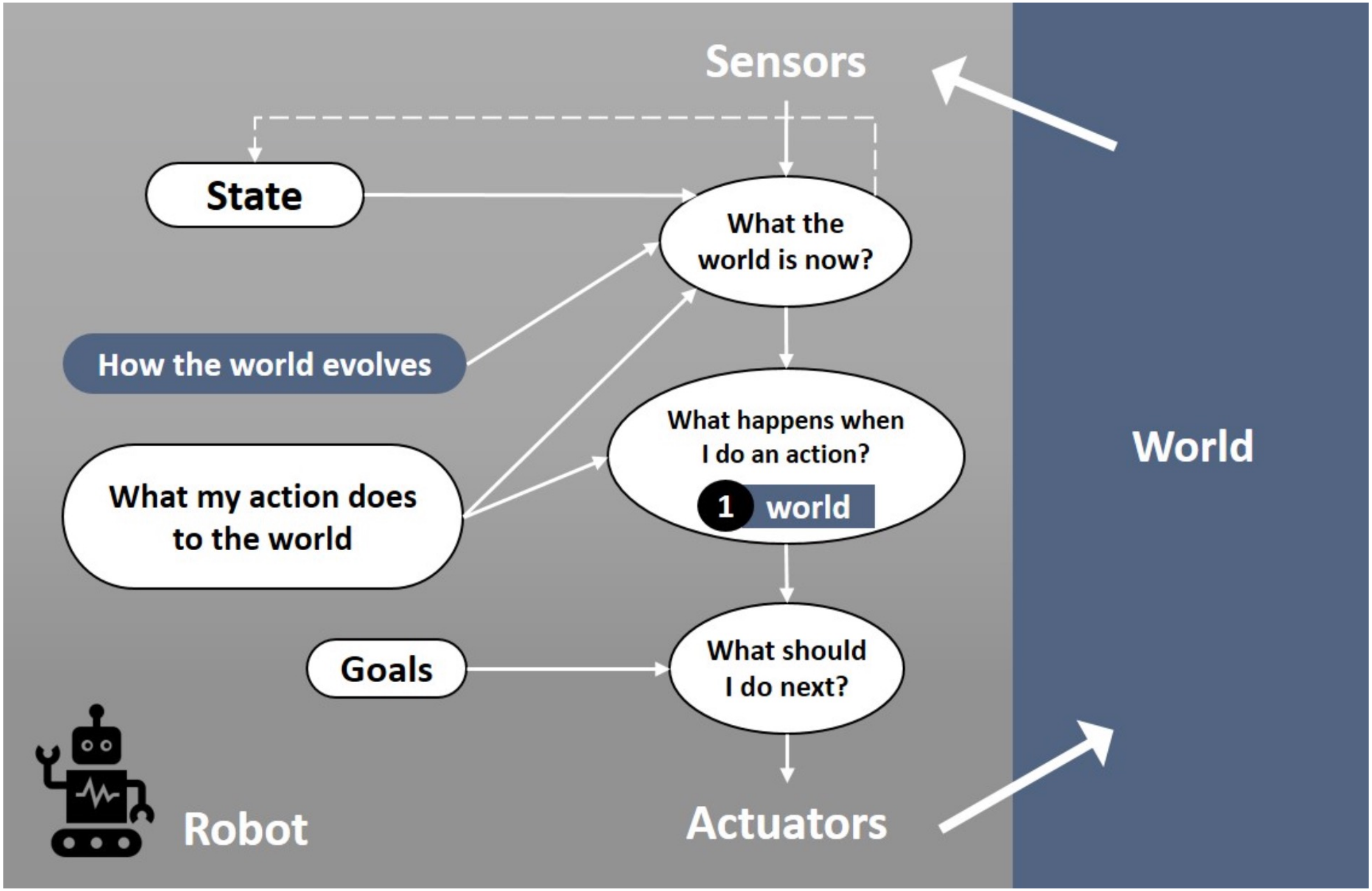
Intelligent Agents
Perception: Language and Vision

Prof. Dr. Ralf Möller
Universität zu Lübeck
Institut für Informationssysteme

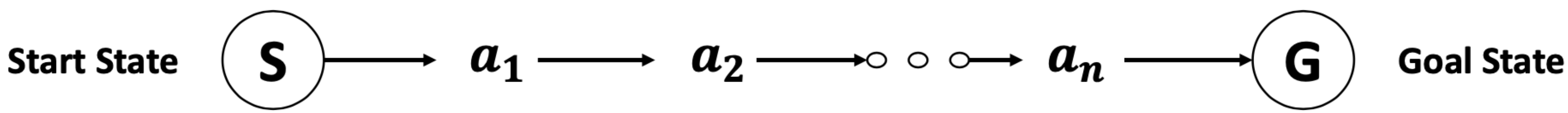
Perception: Agenda (subject to adaptations)

- Perception in intelligent systems for information retrieval (web-mining agent(s))
- (Written) Language
  - Probabilistic dimension reduction, latent content descriptions, topic models, LDA, LDA-HMM
  - Representation learning for sequential structures, embedding spaces, word2vec, CBOW, skip-gram, hierarchical softmax, negative sampling
  - Language models (1d-CNNs. RNNs, LSTMs, ELMo, Transformers, BERT, GPT-4/PaLM 2), Natural language inference and query answering
  - Retrieval, annotation, summarization services (tl;dr)
- Vision (2D-CNNs, Deep Architectures: AlexNet, ResNet)
- Combining language and vision
  - CLIP (OpenAI) / LIT (Google) / data2vec (Facebook) / Flamingo (DeepMind), DALL-E and beyond
- Data structure interpretation
  - Knowledge graph embedding with GNNs, combining embedding-based KG completion with probabilistic graphical models (ExpressGNN, pLogicNet), MLN inference and learning based on embedded knowledge graphs, GMNNs)

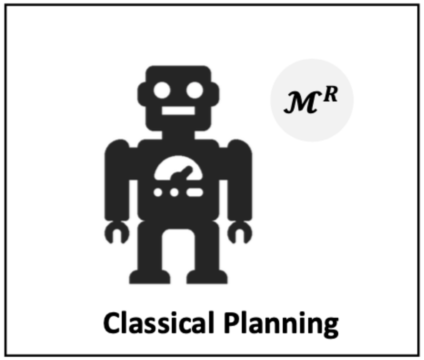
Agents


Sensors
State
What the world is now?
How the world evolves
What happens when I do an action?
1 world
What my action does to the world
Goals
What should I do next?
Robot
Actuators
World

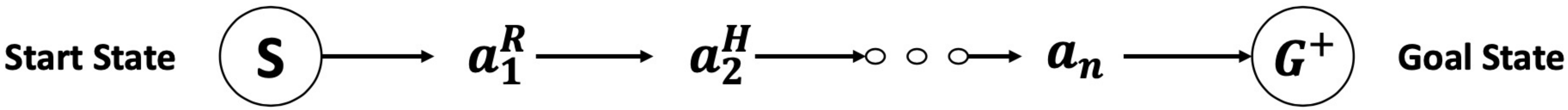
Classical planning



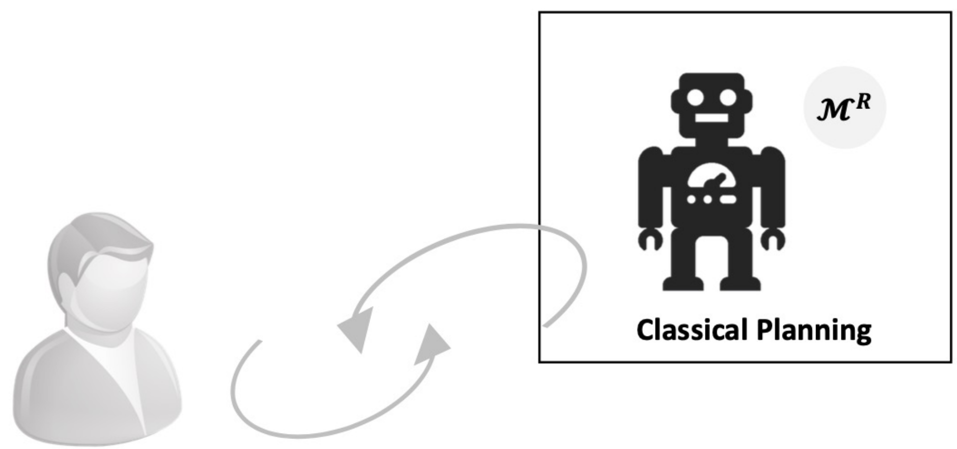
Given – S, G and set of actions $\{a_i\}$ => Agent's Model $\mathbf{M}^R$
*Find – sequence of actions or **plan** $\pi = \langle a_1, a_2, \ldots, a_n \rangle$ that transforms **S** to **G**.*



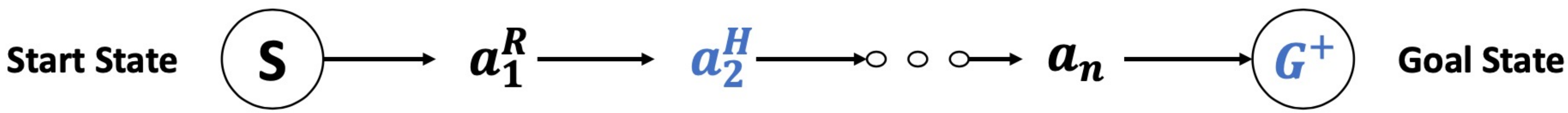
Joint planning



Given – S, G and set of actions $\{a_i\}$ => Agent's Model M^R
*Find – sequence of actions or **joint plan** $\pi = \langle a_1, a_2, \dots, a_n \rangle$ that transforms **S** to G^+.*

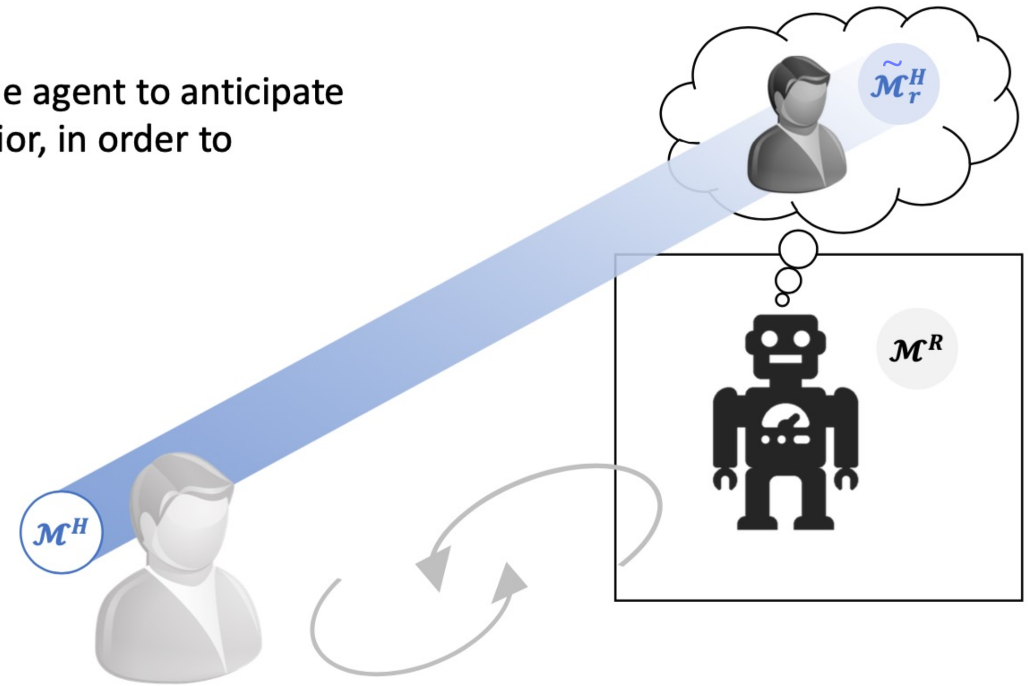


Agent model for human behavior anticipation

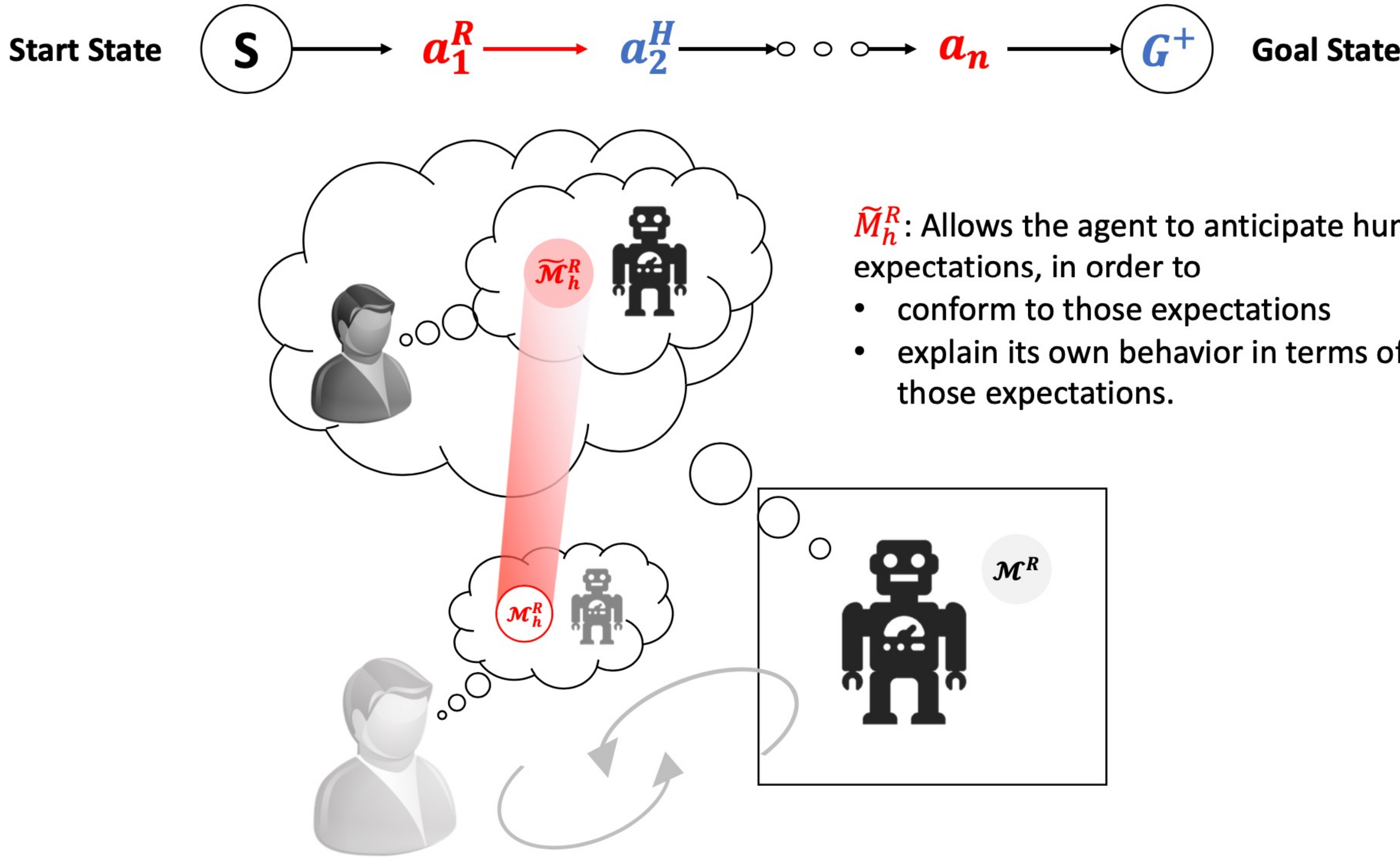


$\widetilde{M}_r^H$: Allows the agent to anticipate human behavior, in order to

- assist
- avoid
- team, etc.



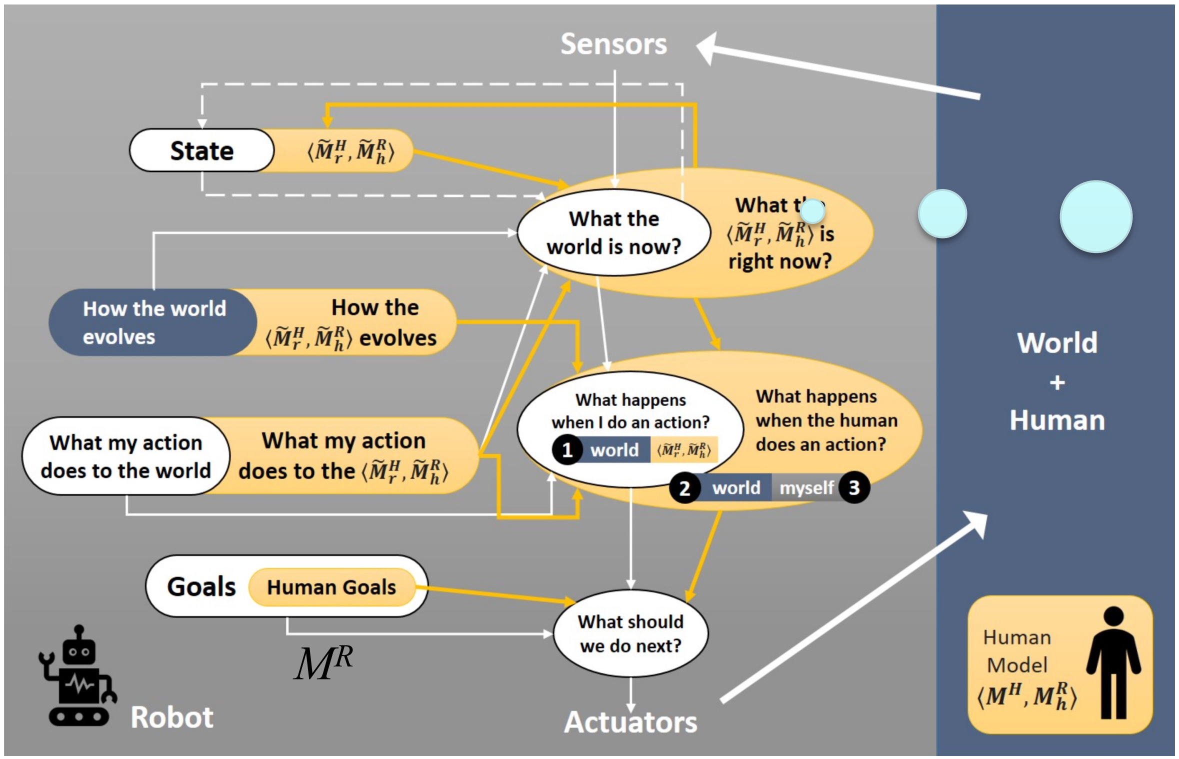
Agent model for human exceptation anticipation



$\widetilde{M}_h^R$: Allows the agent to anticipate human expectations, in order to

- conform to those expectations
- explain its own behavior in terms of those expectations.

Human specifies goal: Solve a certain problem



- M^H human model of the problem to be solved
- $M^R{}_h$ is the human's understanding of the robot's M^R
- M^R robot model of the problem to be solved
- $\tilde{M}^H{}_r$ is the robot's understanding of M^H (anticipate human behavior)
- $\tilde{M}^R{}_h$ is the robot's understanding of $M^R{}_h$ (anticipate human's expectations)

What causes differences in mental models?

- Expectations on capabilities / actions
  - Human may have misconceptions about robot's actions
  - Certain actions in human's mental model may not be feasible for the robot
- Expected state of the world
  - Human may assume certain facts are true in the world (even if they are not true)
- Expected goals
  - Human may have misconceptions about the robot's goals/intentions
  - Robot might need to diagnose this
- Sensor model differences
  - Human may have partial observability of the robot's activities
  - Human may have incorrect beliefs about robot's observational capabilities

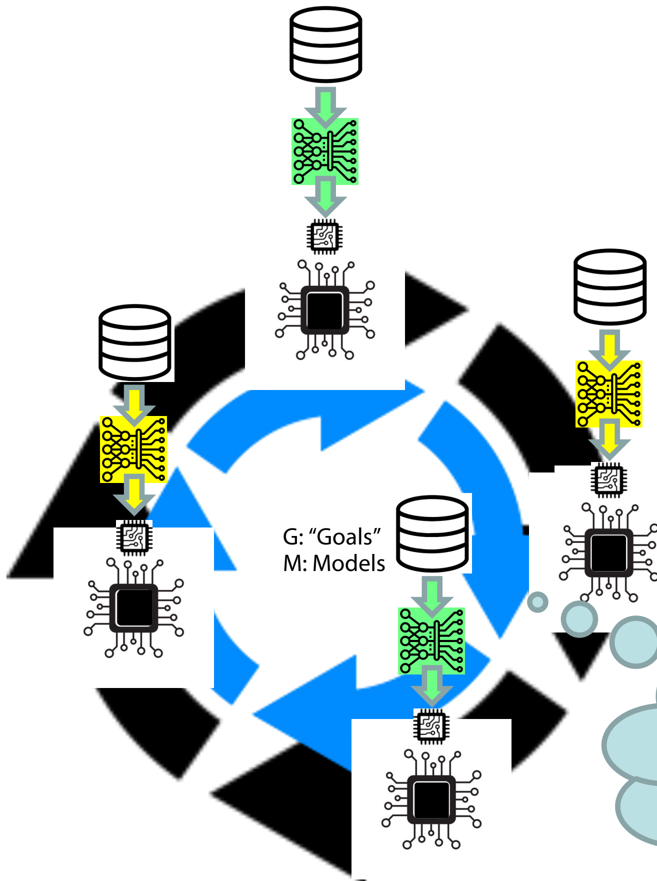
Where do mental models come from?

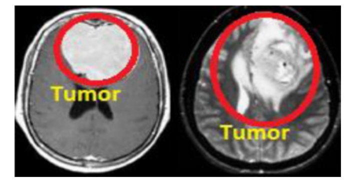
- In certain applications mental models are known beforehand
- Learning simple models for generating explanations/explicability
  - This will be covered later
- Learning full models (transition functions, rewards)
  - Through interaction with users

AI Hypothesis: Agent exhibits intelligent behavior (agere: handeln)

Percepts

Feedback





Action

Explanation

Behavior specification instead of goal

Goal Specification

- For specific problem classes M^H, goal specification languages must be developed to specify M^R
  - If goal specifications cannot sensibly be provided by humans for a certain application domain, AI researchers do need to continue their work
- We will consider …
  - how information retrieval (IR) goals can be represented and communicated to a web-mining agent (IR agent)
  - how uncertainty about IR goals can be reduced
  - how reinforcement feedback can be collected by the IR agent

Recap: Document and query representation

Only represent occurrences of terms with incidence matrix?

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	1	1	0	0	0	1
Brutus	1	1	0	1	0	0
Caesar	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0

Use word counts?

Use word frequencies?

What about terms that occur in certain documents but seldomly in the corpus?

Recap: TF.IDF

f_{ij} = number of terms t_i in document d_j

$$TF_{ij} = f_{ij} \,/\, number\ of\ terms\ in\ d_j$$

n_i = Number of documents with term i

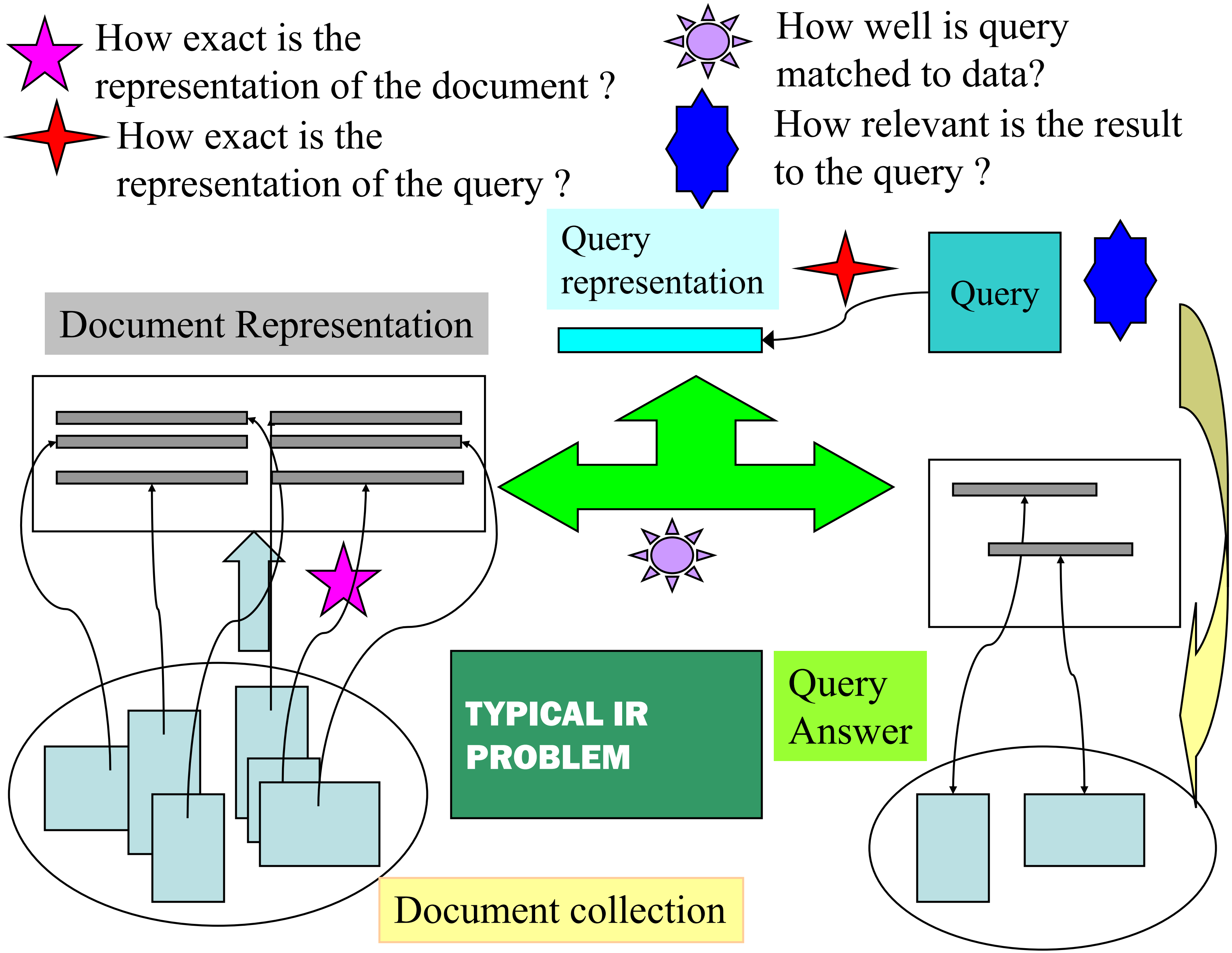
N = Total number of documents

$$IDF_i = \log \frac{N}{n_i}$$

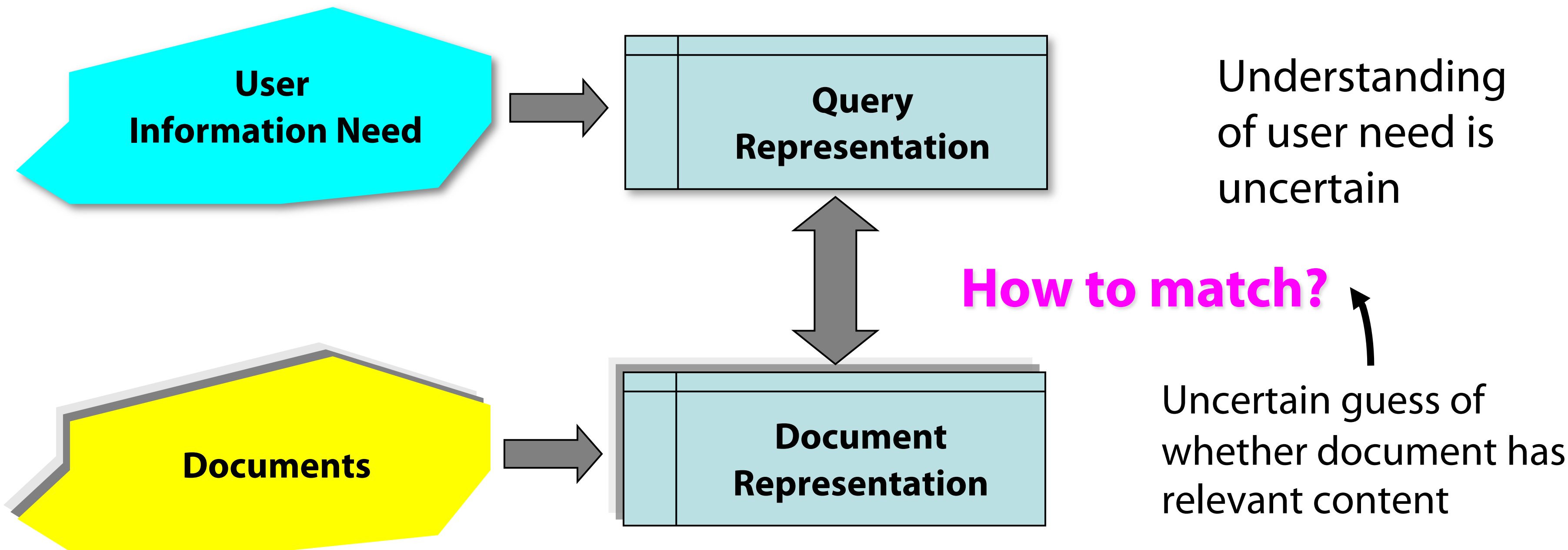
TF.IDF measure

$$w_{ij} = TF_{ij} \cdot IDF_i$$

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	13.1	11.4	0.0	0.0	0.0	0.0
Brutus	3.0	8.3	0.0	1.0	0.0	0.0
Caesar	2.3	2.3	0.0	0.5	0.3	0.3
Calpurnia	0.0	11.2	0.0	0.0	0.0	0.0
Cleopatra	17.7	0.0	0.0	0.0	0.0	0.0
mercy	0.5	0.0	0.7	0.9	0.9	0.3
worser	1.2	0.0	0.6	0.6	0.6	0.0


How exact is the representation of the document ?
How exact is the representation of the query ?
How well is query matched to data?
How relevant is the result to the query ?
Query representation
Query
Document Representation
TYPICAL IR PROBLEM
Query Answer
Document collection

Why probabilities in IR?



In traditional IR systems, matching between each document and query is attempted in a semantically imprecise space of index terms

Probabilities provide a principled foundation for uncertain reasoning
Can we use probabilities to quantify our uncertainties?

Probability Ranking Principle

- Collection of Documents
- User issues a query
- A set of documents is found and needs to be returned
- **Question: In what order to present documents to user ?**
- Need a formal way to judge the "goodness" of documents w.r.t. queries
- **Idea: Probability of relevance of the documents w.r.t. query**

Ben He, Probability Ranking Principle, Reference Work Entry, Encyclopedia of Database Systems, Ling Liu, Tamer, Öszu (Eds.), 2168-2169, Springer, **2009**.

Probabilistic Approaches to IR

- Probability Ranking Principle (Robertson, 70ies; Maron, Kuhns, 1959)

  Robertson S.E. The probability ranking principle in IR. J. Doc., 33:294–304, **1977**.

  M. E. Maron and J. L. Kuhns. On Relevance, Probabilistic Indexing and Information Retrieval. *J. ACM* 7, 3, 216-244, **1960**.

- IR as Probabilistic Inference (van Rijsbergen & et al., since 70ies)

  van Rijsbergen C.J. Inform. Retr.. Butterworths, London, 2nd edn., **1979**.

- Probabilistic IR (Croft, Harper, 70ies)

  Croft W.B. and Harper D.J. Using probabilistic models of document retrieval without relevance information. J. Doc., 35:285–295, **1979**.

- Probabilistic Indexing (Fuhr & et al., late 80ies-90ies)

  Norbert Fuhr. 1989. Models for retrieval with probabilistic indexing. *Inf. Process. Manage.* 25, 1, 55-72, **1989**.

Let us recap probability theory

- Bayesian probability formulas

$$p(a \mid b)\,p(b) = p(a \cap b) = p(b \mid a)\,p(a)$$

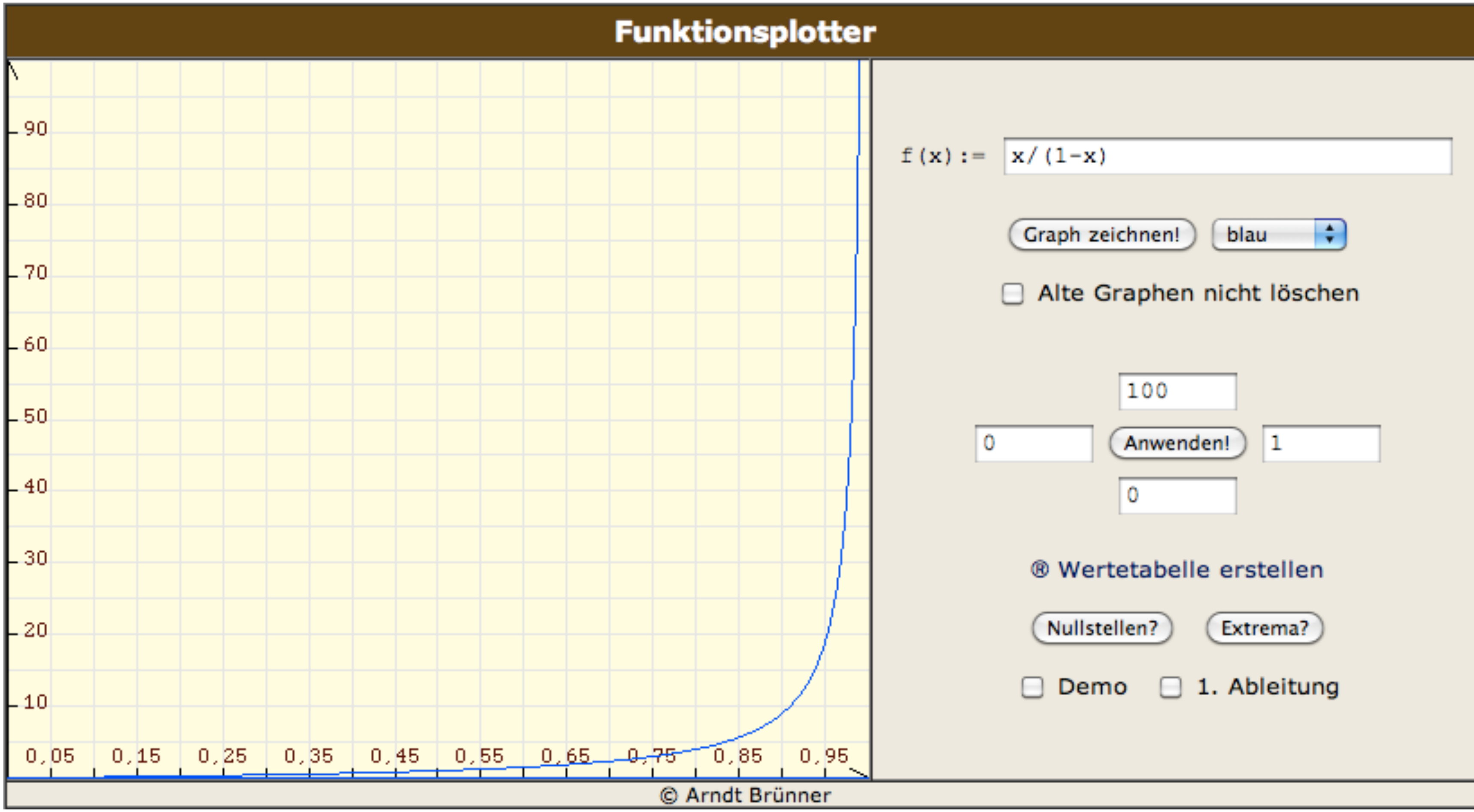
$$p(a \mid b) = \frac{p(b \mid a)\,p(a)}{p(b)}$$

$$p(\overline{a} \mid b)\,p(b) = p(b \mid \overline{a})\,p(\overline{a})$$

- Odds:

$$O(y) = \frac{p(y)}{p(\overline{y})} = \frac{p(y)}{1 - p(y)}$$

Odds vs. Probabilities



Probability Ranking Principle

Let x be a document in the retrieved collection.
Let R represent Relevance=true of a document w.r.t. given (fixed) query and let NR represent Relevance=false.

Need to find p(R|x) - probability that a retrieved document *x* is **relevant.**

$$p(R \mid x) = \frac{p(x \mid R)\,p(R)}{p(x)}$$

$$p(NR \mid x) = \frac{p(x \mid NR)\,p(NR)}{p(x)}$$

p(*R*),p(*NR*) - prior probability of retrieving a relevant or non-relevant document, respectively

p(x|R), p(x|NR) - probability that if a relevant or non-relevant document is retrieved, it is *x*.

van Rijsbergen, Cornelis Joost. Information Retrieval, 2nd edition. Butterworths, **1979**

Probability Ranking Principle

$$p(R \mid x) = \frac{p(x \mid R)\, p(R)}{p(x)}$$

$$p(NR \mid x) = \frac{p(x \mid NR)\, p(NR)}{p(x)}$$

Ranking Principle (Bayes' Decision Rule):

If p(*R*|*x*) > p(*NR*|*x*) then *x* is relevant,

If p(*R*|*x*) ≤ p(*NR*|*x*) then *x* is not relevant

- Note: $p(R \mid x) + p(NR \mid x) = 1$

Probability Ranking Principle

<u>Claim:</u> PRP minimizes the average probability of error

$$p(error \mid x) = \begin{cases} p(R \mid x) & \text{If we decide } \mathbf{NR} \\ p(NR \mid x) & \text{If we decide } \mathbf{R} \end{cases}$$

Expected overall error

$$p(error) = \sum_x p(error \mid x) p(x)$$

p(error) is minimal when all p(error|x) are minimal
Bayes' decision rule minimizes each p(error|x).

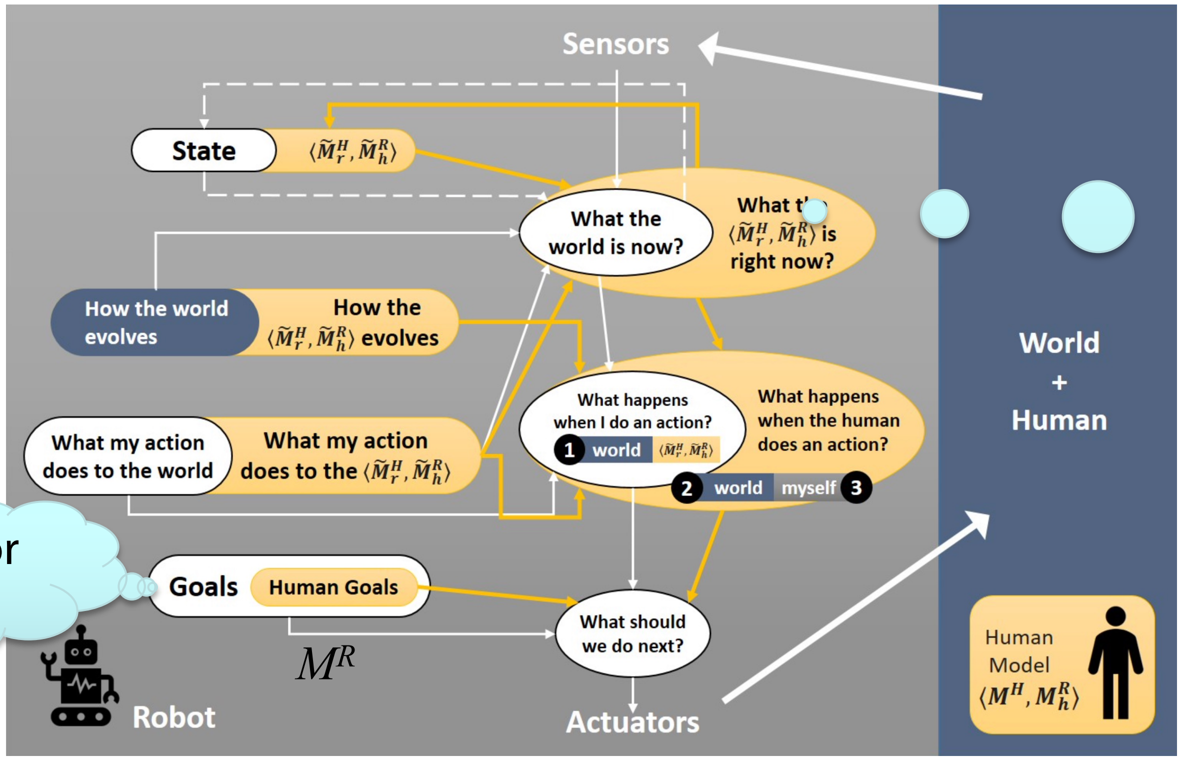
Ranking Principle (Bayes' Decision Rule):
If p(*R*|*x*) > p(*NR*|*x*) then *x* <u>is relevant</u>,
If p(*R*|*x*) ≤ p(*NR*|*x*) then *x* <u>is not relevant</u>

Probability Ranking Principle

- More complex case: retrieval costs
  - *C*: cost of retrieval of relevant document
  - *C′* : cost of retrieval of non-relevant document
  - *d* : a document
- Documents *d* are ranked according to the Probability Ranking Principle when it holds that :

If $C \cdot p(R \mid d) + C' \cdot (1 - p(R \mid d)) \leq C \cdot p(R \mid d') + C' \cdot (1 - p(R \mid d'))$
for any other *d′ not yet retrieved,*
then
d is the next document to be retrieved

Intelligent Autonomous Systems



"Mental models"

Behavior spec

- M^H human model of the problem to be solved
- M^R robot model of the problem to be solved
- $M^R{}_h$ is the human's understanding of the robot's M^R
- $\tilde{M}^H{}_r$ is the robot's understanding of M^H (anticipate human behavior)
- $\tilde{M}^R{}_h$ is the robot's understanding of $M^R{}_h$ (anticipate human's expectations)

Relevance models

- Given: PRP to be applied
  - "Relevance" of each document is independent of relevance of other documents
- Need to estimate probability: $P(R|q,d)$
- Binary Independence Retrieval (BIR):
  - Many documents D - one query q
  - Estimate $P(R|q,d)$ by considering whether $d \in D$ is relevant for q
- Binary Independence Indexing (BII):
  - One document d - many queries Q
  - Estimate $P(R|q,d)$ by considering whether a document d is relevant for a query $q \in Q$

Binary Independence Retrieval

- **"Binary" = Boolean**: documents are represented as binary vectors of terms:
  - $\vec{x} = (x_1, \ldots, x_n)$
  - $x_i = 1$ iff term *i* is present in document *x*.
- **"Independence":** terms occur in documents independently
- Different documents can be modeled as same vector.

Binary Independence Retrieval

- Queries: binary vectors of terms
- Given query q,
  - for each document d need to compute p(Relevant=true|q,d)
  - replace with computing **p(Relevant=true|q,x)** where **x** is vector representing **d**
- Interested only in ranking
- Will use odds (the higher, the better):

$$O(R \mid q, \vec{x}) = \frac{p(R \mid q, \vec{x})}{p(NR \mid q, \vec{x})} = \frac{p(R \mid q)}{p(NR \mid q)} \cdot \frac{p(\vec{x} \mid R, q)}{p(\vec{x} \mid NR, q)}$$

Binary Independence Retrieval

$$O(R \mid q, \vec{x}) = \frac{p(R \mid q, \vec{x})}{p(NR \mid q, \vec{x})} = \frac{p(R \mid q)}{p(NR \mid q)} \cdot \frac{p(\vec{x} \mid R, q)}{p(\vec{x} \mid NR, q)}$$

Constant for each query

Needs estimation

- Using **Independence** Assumption:

$$\frac{p(\vec{x} \mid R, q)}{p(\vec{x} \mid NR, q)} = \prod_{i=1}^{n} \frac{p(x_i \mid R, q)}{p(x_i \mid NR, q)}$$

- So : $O(R \mid q, d) = O(R \mid q) \cdot \prod_{i=1}^{n} \frac{p(x_i \mid R, q)}{p(x_i \mid NR, q)}$

Binary Independence Retrieval

$$O(R \mid q,d) = O(R \mid q) \cdot \prod_{i=1}^{n} \frac{p(x_i \mid R,q)}{p(x_i \mid NR,q)}$$

- Since x_i is either 0 or 1:

$$O(R \mid q,d) = O(R \mid q) \cdot \prod_{x_i=1} \frac{p(x_i=1 \mid R,q)}{p(x_i=1 \mid NR,q)} \cdot \prod_{x_i=0} \frac{p(x_i=0 \mid R,q)}{p(x_i=0 \mid NR,q)}$$

- Let $p_i = p(x_i = 1 \mid R,q); \quad r_i = p(x_i = 1 \mid NR,q);$

- Assume, for all terms not occurring in the query ($q_i=0$) $p_i = r_i$

Then...

Binary Independence Retrieval

$$O(R \mid q, \vec{x}) = O(R \mid q) \cdot \prod_{x_i = q_i = 1} \frac{p_i}{r_i} \cdot \prod_{\substack{x_i = 0 \\ q_i = 1}} \frac{1 - p_i}{1 - r_i}$$

All matching terms

Non-matching query terms (too many)

-

All matching terms

All query terms

Binary Independence Retrieval

$$O(R \mid q, \vec{x}) = O(R \mid q) \cdot \prod_{x_i = q_i = 1} \frac{p_i(1 - r_i)}{r_i(1 - p_i)} \cdot \prod_{q_i = 1} \frac{1 - p_i}{1 - r_i}$$

Constant for each query

Only quantity to be estimated for rankings

- Optimize Retrieval Status Value (RSV):

$$RSV = \log \prod_{x_i = q_i = 1} \frac{p_i(1 - r_i)}{r_i(1 - p_i)} = \sum_{x_i = q_i = 1} \log \frac{p_i(1 - r_i)}{r_i(1 - p_i)}$$

Binary Independence Retrieval

- All boils down to computing RSV.

$$RSV = \log \prod_{x_i = q_i = 1} \frac{p_i(1-r_i)}{r_i(1-p_i)} = \sum_{x_i = q_i = 1} \log \frac{p_i(1-r_i)}{r_i(1-p_i)}$$

$$RSV = \sum_{x_i = q_i = 1} c_i;$$

For all query terms i:
Find docs containing term i (→ inverted index)

Nonstandard Databases and Data Mining

$$c_i = \log \frac{p_i(1-r_i)}{r_i(1-p_i)} = \log \frac{p_i}{(1-p_i)} + \log \frac{(1-r_i)}{r_i}$$

So, how do we compute ci's from our data ?

Binary Independence Retrieval

- Estimating RSV coefficients: Groundtruth for subset of docs
  - It is known wether docs are relevant or not
- For each term i look at the following table:

Document	Relevant	Non-Relevant	Total
$X_i=1$	s	$n-s$	n
$X_i=0$	$S-s$	$N-n-S+s$	$N-n$
Total	S	$N-S$	N

$$p_i = p(x_i = 1 \mid R, q); \quad r_i = p(x_i = 1 \mid NR, q);$$

- Estimates: $p_i \approx \frac{s}{S} \qquad r_i \approx \frac{(n-s)}{(N-S)}$

Binary Independence Retrieval

- Estimating RSV coefficients.
- For each term *i* look at the following table:

Document	Relevant	Non-Relevant	Total
$X_i=1$	s	$n-s$	n
$X_i=0$	$S-s$	$N-n-S+s$	$N-n$
Total	S	$N-S$	N

- Estimates: $p_i \approx \frac{s}{S}$ $\quad r_i \approx \frac{(n-s)}{(N-S)}$ $\quad c_i = \log \frac{p_i(1-r_i)}{r_i(1-p_i)}$

$$c_i \approx K(N,n,S,s) = \log \frac{s/(S-s)}{(n-s)/(N-n-S+s)}$$

Avoid division by 0

$$c_i \approx K(N,n,S,s) = \log \frac{(s+1/2)\big/(S-s+1/2)}{(n-s+1/2)\big/(N-n-S+s+1/2)}$$

Estimation in practice

$$p_i \approx \frac{s}{S} \qquad r_i \approx \frac{(n-s)}{(N-S)} \qquad c_i = \log\frac{p_i}{(1-p_i)} + \log\frac{(1-r_i)}{r_i}$$

- If non-relevant documents are approximated by the whole collection (S=s=0), then r_i (prob. of occurrence term i in non-relevant documents for query) is n/N and
  - $\log (1-r_i)/r_i = \log (N-n)/n \approx \log(1+(N-n)/n) = \log N/n = IDF$
- Idea cannot be easily extended to p_i
- Estimate p_i (probability of occurrence of term i in relevant docs):
  - From relevant documents if we know some
  - Use constant 0.5 – then just get idf weighting of terms (p_i and $1-p_i$ cancel out)
  - …

$$RSV = \sum_{x_i=q_i=1} \log\frac{N}{n_i}$$

- We have a nice theoretical foundation of TF.IDF (in the binary case: TF=1 or TF=0)

Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. In *Document retrieval systems*, Vol. 3. Taylor Graham Publishing, London, UK, UK 132-142. **1988**.

Greiff, Warren R., A Theory of Term Weighting Based on Exploratory Data Analysis. In: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 11-19, **1998**.

Robertson S.E., Understanding inverse document frequency: On theoretical arguments for idf. J. Doc., 60:503–520, **2004**.

Iteratively estimating p_i

Expectation Maximization:

1. Assume that p_i constant over all q_i in query
   - $p_i = 0.5$ (even odds) for any given doc
2. Determine guess of relevant document set from subset V:
   - V is fixed size set of highest ranked documents on this model
3. We need to improve our guesses for p_i and r_i, so
   - Use distribution of q_i in docs in V. Let V_i be set of documents containing q_i
     - $p_i = |V_i| / |V|$
   - Assume if not retrieved then not relevant
     - $r_i = (n_i - |V_i|) / (N - |V|)$
4. Go to 2. until convergence then return ranking

Probabilistic Relevance Feedback

1. Guess a preliminary probabilistic description of *R* and use it to retrieve a first set of documents V, as above.
2. Interact with the user to refine the description: learn some definite members of R and NR
3. Reestimate p_i and r_i on the basis of these
   - Or can combine new information with original guess (use Bayesian prior):

$$p_i^{(2)} = \frac{|V_i| + \lambda p_i^{(1)}}{|V| + \lambda}$$

λ is prior weight

4. Repeat, thus generating a succession of approximations to *R*.

Binary Independence Indexing

- "Learning" from queries
  - More queries: better results

$$p(R \mid \vec{q}, \vec{x}) = \frac{p(\vec{q} \mid \vec{x}, R)\, p(R \mid \vec{x})}{p(\vec{q} \mid \vec{x})}$$

- **p(q|x,R)** - probability that if document **x** had been deemed relevant, query **q** had been asked
- The rest of the framework is similar to BIR

Recap

- Agents have goals
  - Example: provide a good IR service
- Goals reflected as utilities
  - If IR is good, agent's utility is maximized

- Our goal: Understand IR principles
  - Mathematical foundations of IR (e.g., PRP)
  - IR quality measures
  - …

- These insights motivate the content of the course

Binary Independence Retrieval vs. Binary Independence Indexing

BIR

- Many Documents,
  One Query
- Bayesian Probability:

$$p(R \mid \vec{q}, \vec{x}) = \frac{p(\vec{x} \mid \vec{q}, R)\, p(R \mid \vec{q})}{p(\vec{x} \mid \vec{q})}$$

- Varies: document representation
- Constant: query (representation)

BII

- One Document,
  Many Queries
- Bayesian Probability

$$p(R \mid \vec{q}, \vec{x}) = \frac{p(\vec{q} \mid \vec{x}, R)\, p(R \mid \vec{x})}{p(\vec{q} \mid \vec{x})}$$

- Varies: query
- Constant: document

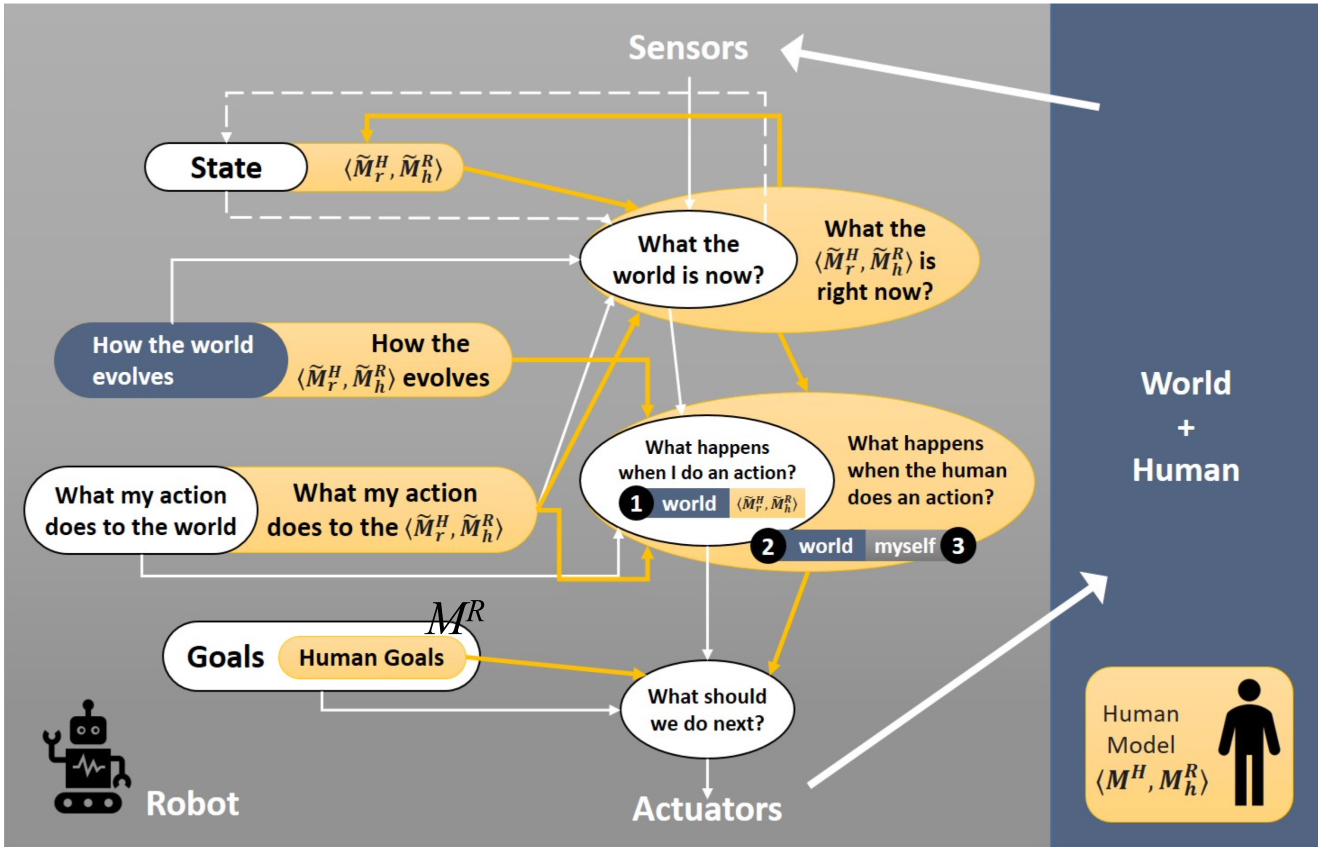
PRP and BIR/BII: The lessons

- Getting reasonable approximations of probabilities is possible.
- Simple methods work only with restrictive assumptions:
  - **term independence**
  - **terms not in query do not affect the outcome**
  - **boolean representation of documents/queries**
  - **document relevance values are independent**
- Some of these assumptions can be removed

Summary: Probabilistic Information Retrieval

- PRP defines a well-defined framework for IR
  - Can understand pragmatic approaches (e.g., TF.IDF)
  - Can be used for formalizing IR (What is the IR problem?)
  - Provides for means to compute ranking of results
- Agents can use different models and different QA strategies for IR
- We will see soon:
  - Agents can update internal models by reinforcement feedback
  - Agents can adapt strategies to new user queries (new goals to be expected)

Recap: Agent architecture $M^R = IR(Q)$



Action:
Return retrieved docs

Waiting for feedback (possibly with follow-up queries)

Retrieved docs to be evaluated by human

Confusion Matrix (e.g., for Classification)

In the example confusion matrix below, of the 8 actual cats, the system predicted that three were dogs, and of the six dogs, it predicted that one was a rabbit and two were cats. We can see from the matrix that the system in question has trouble distinguishing between cats and dogs, but can make the distinction between rabbits and other types of animals pretty well.

Example confusion matrix

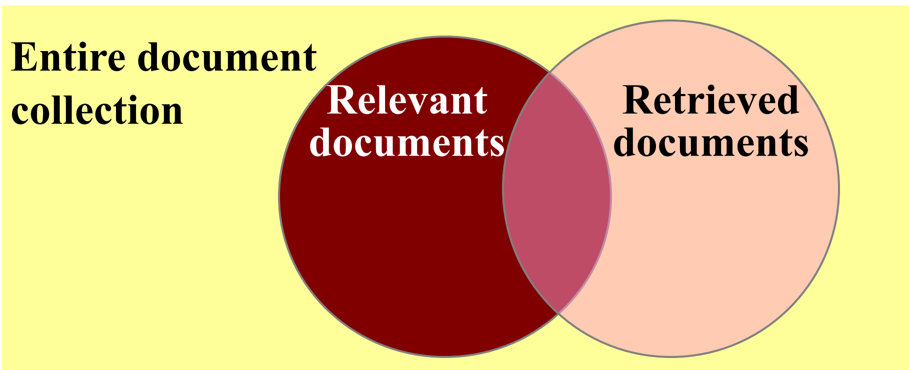
	Cat	**Dog**	**Rabbit**
Cat	5	3	0
Dog	2	3	1
Rabbit	0	2	11

Understanding where an agent has deficiencies

(Direct) feedback:
Present confusion matrix to an agent

Reinforcement:
Relevance feedback for retrieval results
(agent might build confusion matrix internally)

Unranked retrieval evaluation: Precision and Recall



	retrieved	not retrieved
irrelevant	retrieved & irrelevant	Not retrieved & irrelevant
relevant	retrieved & relevant	not retrieved but relevant

$$recall = \frac{Number\ of\ relevant\ documents\ retrieved}{Total\ number\ of\ relevant\ documents}$$

$$precision = \frac{Number\ of\ relevant\ documents\ retrieved}{Total\ number\ of\ documents\ retrieved}$$

Unranked retrieval evaluation: Precision and Recall

- Precision: fraction of retrieved docs that are relevant = P(retr & rel | retrieved)
- Recall: fraction of relevant docs that are retrieved = P(retr & rel | relevant in repos)

	Relevant	Not Relevant
Retrieved	true positives (tp)	false positives (fp)
Not Retrieved	false negatives (fn)	true negatives (tn)

- Precision = tp/(tp + fp)
- Recall = tp/(tp + fn)

Difficult to optimize both indicators at the same time

Overview on evaluation measures

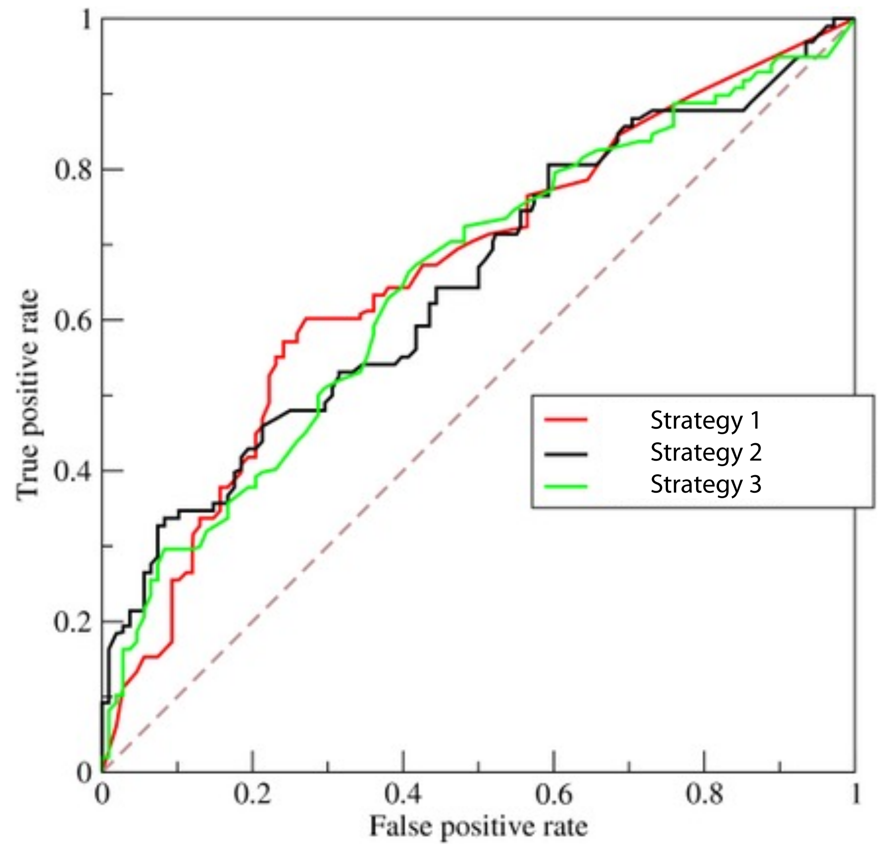
		True condition			
	Total population	Condition positive	Condition negative	Prevalence $= \frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	
Predicted condition	Predicted condition positive	**True positive**	**False positive** (Type I error)	Positive predictive value (PPV), Precision $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Test outcome positive}}$	False discovery rate (FDR) $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Test outcome positive}}$
	Predicted condition negative	**False negative** (Type II error)	**True negative**	False omission rate (FOR) $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Test outcome negative}}$	Negative predictive value (NPV) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Test outcome negative}}$
	Accuracy (ACC) = $\frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$	True positive rate (TPR), Sensitivity, Recall $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR), Fall-out $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	Positive likelihood ratio (LR+) $= \frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) $= \frac{\text{LR+}}{\text{LR−}}$
		False negative rate (FNR), Miss rate $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	True negative rate (TNR), Specificity (SPC) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$	Negative likelihood ratio (LR–) $= \frac{\text{FNR}}{\text{TNR}}$	

[Wikipedia]

https://en.wikipedia.org/wiki/Evaluation_measures_(information_retrieval)#Mean_average_precision

Relative operating characteristic (ROC)

- What if goal specification involves control parameters?
- E.g., for strategies
- Investigate effects of parameter adjustments
- Compare TP rate (recall) and FP rate (fall-out)
- Example w/ three strategies
- Measure:
  Area under curve (AUC)
  curve = ROC



Back to Precision and Recall

- Determining Recall can be difficult
- Total number of relevant items is sometimes not available – use **pooling**
  - Sample across the database and perform relevance judgment on these items
  - Apply different retrieval algorithms to the same database for the same query. The aggregate of relevant items is taken as the total relevant set

First instruct agent to do IR on known corpus (train agent to use best QA strategy)
Then send it out.

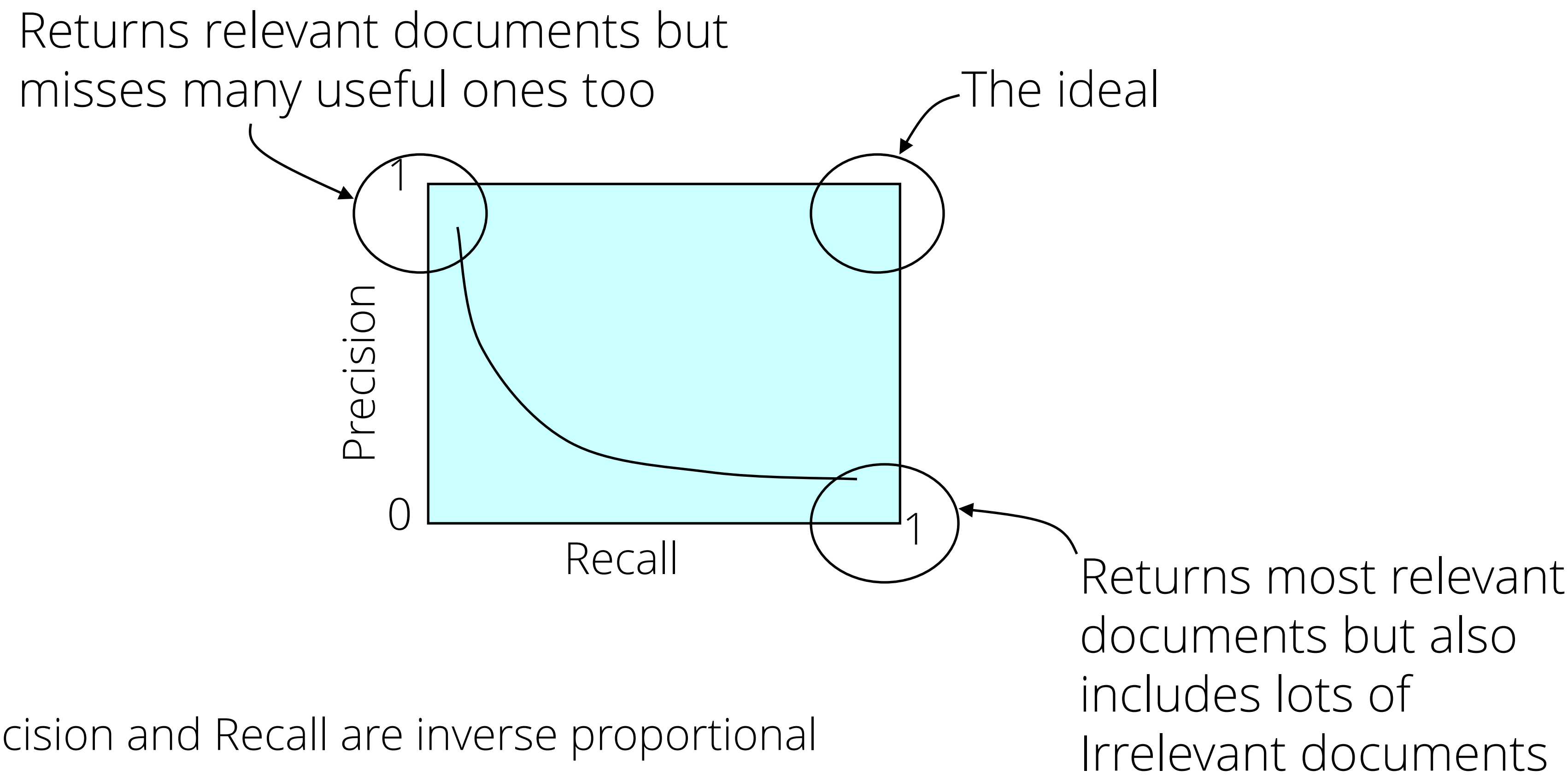
Standard Methodology for Measuring Relevance in IR

- To measure relevance effectiveness of ad-hoc IR, we need:
  1. A **document collection**
  2. A suite of information needs, expressible as **queries**
     - Must be representative of actual user needs
     - Sample from query logs, if available
  3. **Binary assessments** of either Relevant or Nonrelevant for each query and each document
     - Can be more nuanced, e.g., 0, 1, 2, 3, …
     - Use *pooling*, when it is unfeasible to assess every (*q*, *d*) pair

The TREC Benchmark

- TREC: Text REtrieval Conference (http://trec.nist.gov/)
  - Became an annual conference in 1992, co-sponsored by the National Institute of Standards and Technology (NIST) and DARPA.
  - Participants are given parts of a standard set of documents and TOPICS (from which queries have to be derived) in different stages for training and testing.
  - Participants submit the P/R values for the final document and query corpus and present their results at the conference.

Trade-off between Recall and Precision



Precision and Recall are inverse proportional

F-measure

- One measure of performance that takes into account both recall and precision
- Precision (P) and Recall (R) are rates
- Harmonic mean of recall and precision:

$$F = \frac{2PR}{P+R} = \frac{2}{\frac{1}{R}+\frac{1}{P}}$$

Positive predictive value (PPV), Precision

$$= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Test outcome positive}}$$

True positive rate (TPR), Sensitivity, Recall

$$= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$$

- In contrast to arithmetic mean, both need to be high for harmonic mean to be high

Ranked Retrieval Measures

- Binary relevance:
  - 11-point Interpolated Precision-Recall Curve
  - R-precision
  - Precision@K (P@K) and Recall@K (R@K)
  - Mean Average Precision (MAP)

Recall-Precision Curves: An Example

n	doc #	relevant
1	588	x
2	589	x
3	576	
4	590	x
5	986	
6	592	x
7	984	
8	988	
9	578	
10	985	
11	103	
12	591	
13	772	x
14	990	

Let total # of relevant docs = 6
Check each new recall point:

R=1/6=0.167;P=1/1=1

R=2/6=0.333;P=2/2=1

R=3/6=0.5; P=3/4=0.75

R=4/6=0.667; P=4/6=0.667

R=5/6=0.833;p=5/13=0.38

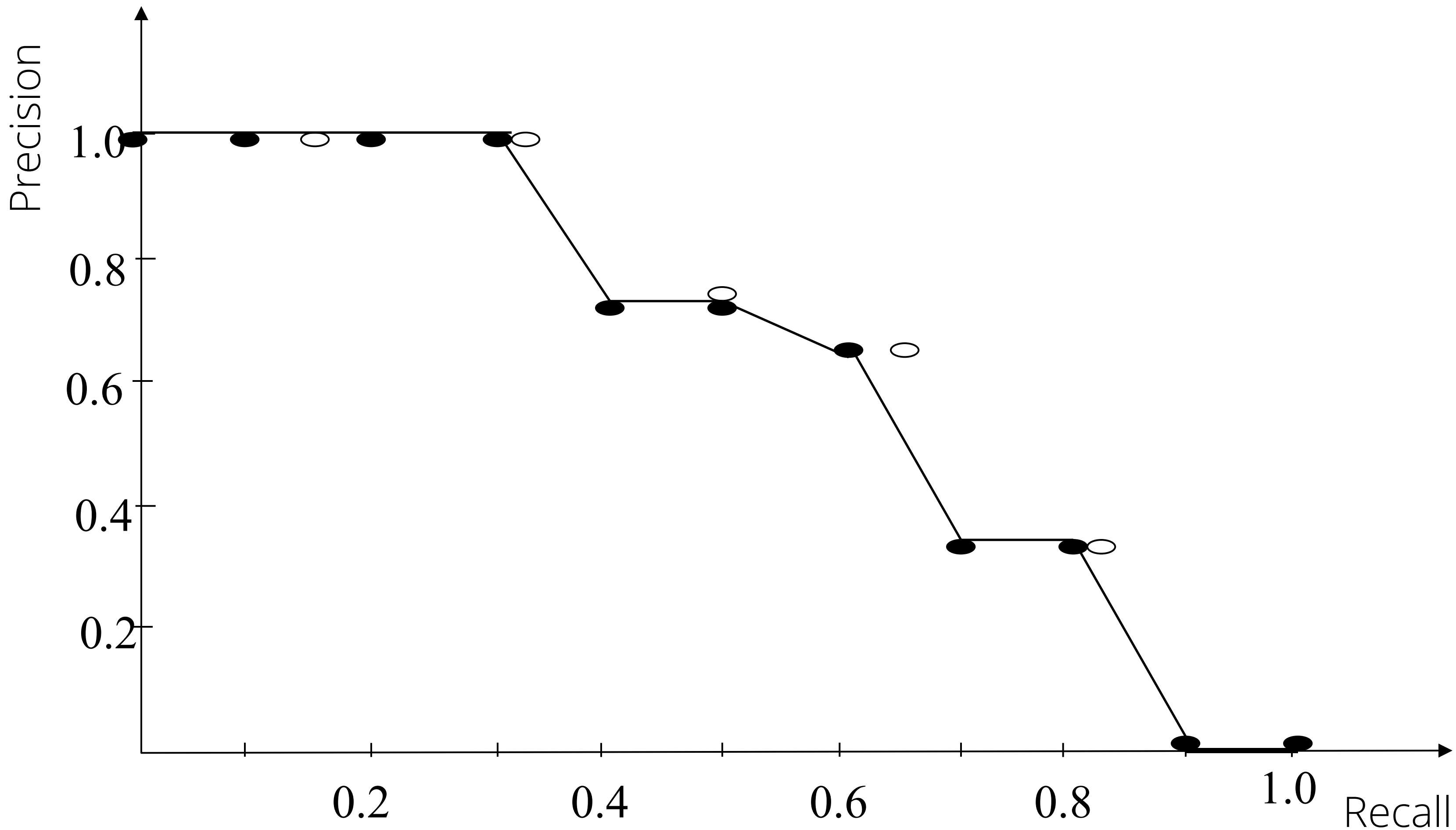
Missing one relevant document: Never reach 100% recall

Interpolating a Recall/Precision Curve

- Interpolate a precision value for each *standard recall level*:
  - $r_j \in \{0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$
  - $r_0 = 0.0, r_1 = 0.1, \ldots, r_{10} = 1.0$
- The interpolated precision at the *j*-th standard recall level is the maximum known precision at any recall level between the *j*-th and (*j* + 1)-th level:

$$P(r_j) = \max_{r_j \le r \le r_{j+1}} P(r)$$

Interpolating a Recall/Precision Curve: An Example



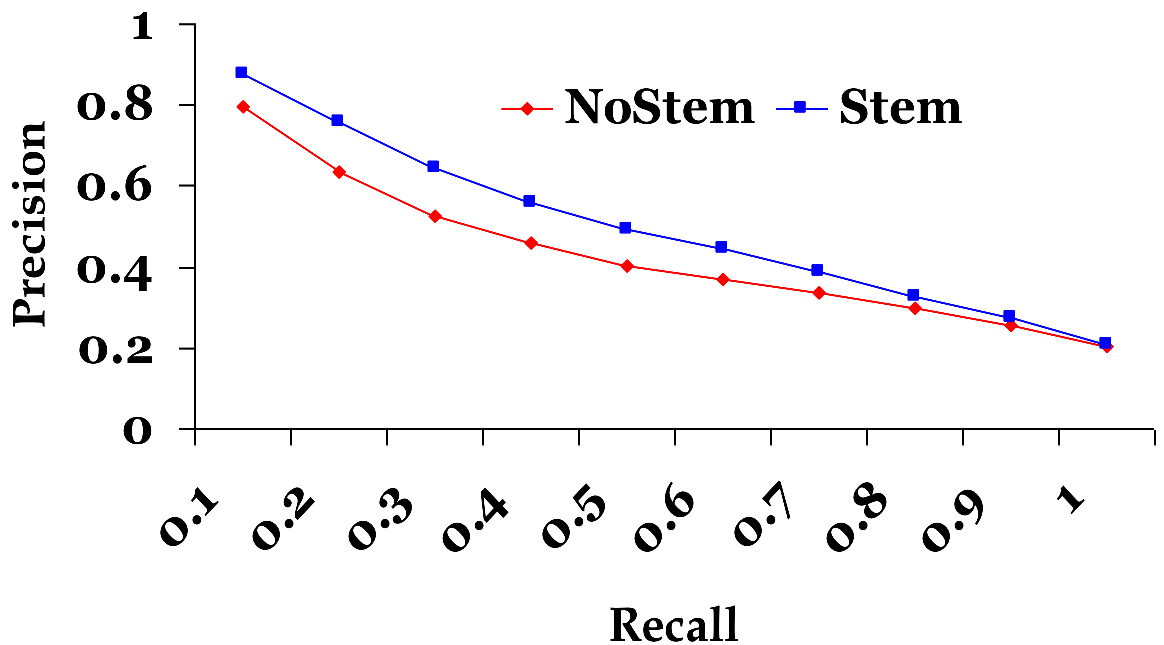
Average Recall/Precision Curve

- Typically average performance over a large ***set*** of queries.
- Compute average precision at each standard recall level across all queries.
- Plot average precision/recall curves to evaluate overall system performance on a document/query corpus.

- Average:
  - Micro-average: compute P/R/F once for the entire set of queries
  - Macro-average: average of within-query precision/recall

How To Compare Two or More Systems

- The curve closest to the upper right-hand corner of the graph indicates the best performance



R-precision

- Precision at the R-th position in the ranking of results for a query that has R relevant documents.

n	doc #	relevant
1	588	x
2	589	x
3	576	
4	590	x
5	986	
6	592	x
7	984	
8	988	
9	578	
10	985	
11	103	
12	591	
13	772	x
14	990	

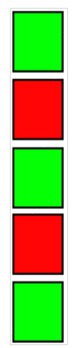
R = # of relevant docs = 6

R-Precision = 4/6 = 0.67

Precision@K

1. Set a rank threshold K.
2. Compute % of documents relevant in top K.
   - Ignores documents ranked lower than K.

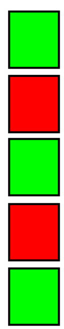
- Example:
  - Prec@3 of 2/3
  - Prec@4 of 2/4
  - Prec@5 of 3/5



- In a similar way we have Recall@K

Mean Average Precision (MAP)

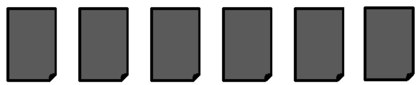
1. Consider rank position of each of the R relevant docs:
   – $K_1, K_2, \ldots K_R$
2. Compute Precision@K for each $K_1, K_2, \ldots K_R$.
3. Average precision = average of P@K.

Example:  has AvgPrec of

$$\frac{1}{3} \cdot \left(\frac{1}{1} + \frac{2}{3} + \frac{3}{5} \right) \approx 0.76$$

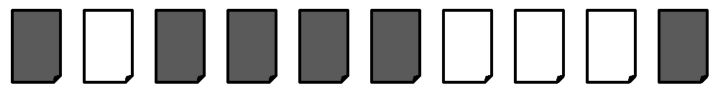
- MAP is Average Precision across multiple queries.

Average Precision for Comparing Rankings



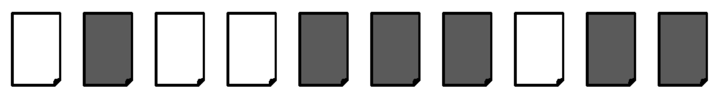
= the relevant documents

Ranking #1



Recall	0.17	0.17	0.33	0.5	0.67	0.83	0.83	0.83	0.83	1.0
Precision	1.0	0.5	0.67	0.75	0.8	0.83	0.71	0.63	0.56	0.6

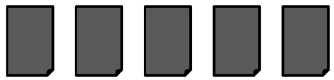
Ranking #2

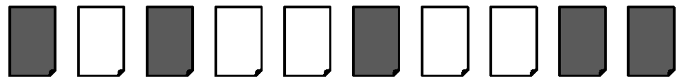


Recall	0.0	0.17	0.17	0.17	0.33	0.5	0.67	0.67	0.83	1.0
Precision	0.0	0.5	0.33	0.25	0.4	0.5	0.57	0.5	0.56	0.6

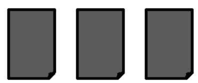
Ranking #1 = (1.0+0.67+0.75+0.8+0.83+0.6) /6 = 0.78
Ranking #2 = (0.5+0.4+0.5+0.57+0.56+0.6) /6 = 0.5

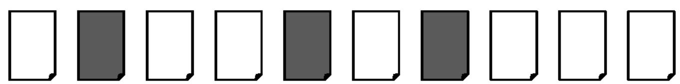
Mean Average Precision (MAP)

 = relevant documents for query 1

Ranking #1 

Recall	0.2	0.2	0.4	0.4	0.4	0.6	0.6	0.6	0.8	1.0
Precision	1.0	0.5	0.67	0.5	0.4	0.5	0.43	0.38	0.44	0.5

 = relevant documents for query 2

Ranking #2 

Recall	0.0	0.33	0.33	0.33	0.67	0.67	1.0	1.0	1.0	1.0
Precision	0.0	0.5	0.33	0.25	0.4	0.33	0.43	0.38	0.33	0.3

Average precision query 1 = (1.0+0.67+0.5+0.44+0.5)/5 = 0.62
Average precision query 2 = (0.5+0.4+0.43)/3 = 0.44
MAP = (0.62 + 0.44)/2 = 0.53

Mean Average Precision (MAP)

- If a relevant document never gets retrieved, we assume the precision corresponding to that relevant document to be zero.
- MAP is macro-averaging: each query counts equally.
- A commonly used measure in current IR research, along with P/R/F

Collaboration: Measure for inter-judge (dis)agreement

- Kappa measure
  - (Dis)Agreement measure among judges
  - Designed for categorical judgments
  - Corrects for chance agreement
- κ = [P(A) – P(E)] / [1 – P(E)]
- P(A) – proportion of time judges agree (observed)
- P(E) – what agreement would be by chance (hypothetical)
- κ = 0 for chance agreement, 1 for total agreement

- In statistics many other measures are defined

Cohen, Jacob, "A coefficient of agreement for nominal scales". *Educational and Psychological Measurement* **20** (1): 37–46, **1960**

Kappa Measure: Example

$P(A)?\ P(E)?$

Number of docs	Judge 1	Judge 2
300	Relevant	Relevant
70	Nonrelevant	Nonrelevant
20	Relevant	Nonrelevant
10	Nonrelevant	Relevant

Kappa Example

- P(A) = 370/400 = 0.925
- P(nonrelevant) = (10+20+70+70)/800 = 0.2125
- P(relevant) = (10+20+300+300)/800 = 0.7878
- P(E) = $0.2125^2 + 0.7878^2$ = 0.665
- κ = (0.925 – 0.665)/(1-0.665) = 0.776

- $\kappa > 0.8$ = good agreement
- $0.67 < \kappa < 0.8$ -> "tentative conclusions"
- Depends on purpose of study
- For >2 judges: average pairwise κs

Carletta, J. C., Assessing agreement on classification tasks:
The kappa statistic. Computational Linguistics, 22(2), 249-254, **1996**

Relevance Feedback: Rocchio Algorithm

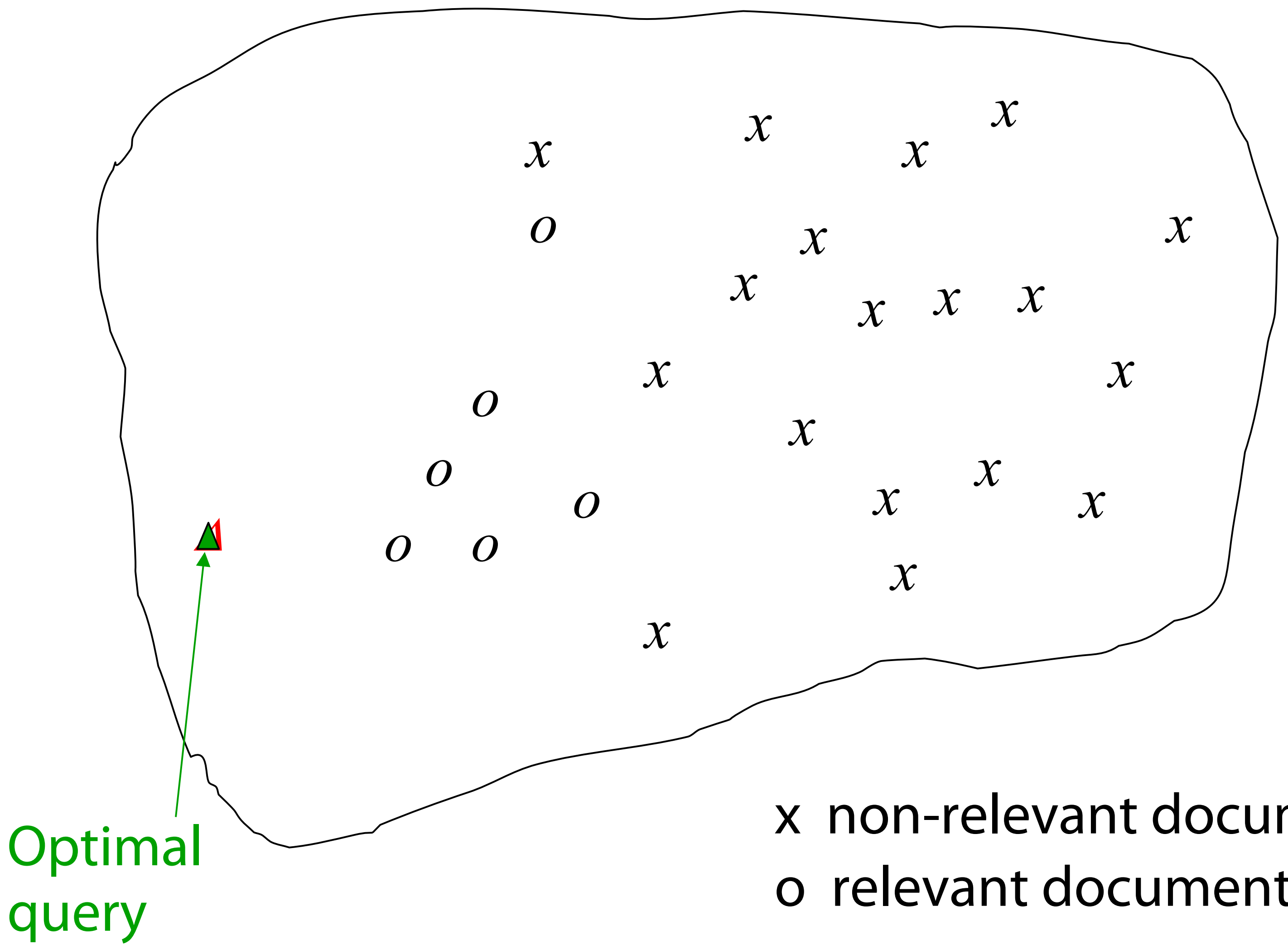
- The Rocchio algorithm incorporates relevance feedback information into the vector space model.
- Want to maximize $|sim(Q, C_r) - sim(Q, C_{nr})|$ where C_r and C_{nr} denote relevant and non-relevant doc vectors, respectively
- The optimal query vector for separating relevant and non-relevant documents (with cosine sim.):

$$\vec{Q}_{opt} = \frac{1}{|C_r|} \sum_{\vec{d}_j \in C_r} \vec{d}_j - \frac{1}{N - |C_r|} \sum_{\vec{d}_j \notin C_r} \vec{d}_j$$

Q_{opt} = optimal query; C_r = set of rel. doc vectors in corpus; N = collection size

- Unrealistic definition:
  We don't know relevant documents in corpus

The Theoretically Best Query



x non-relevant documents
o relevant documents

Rocchio 1971 Algorithm (SMART System)

- Useful in practice:

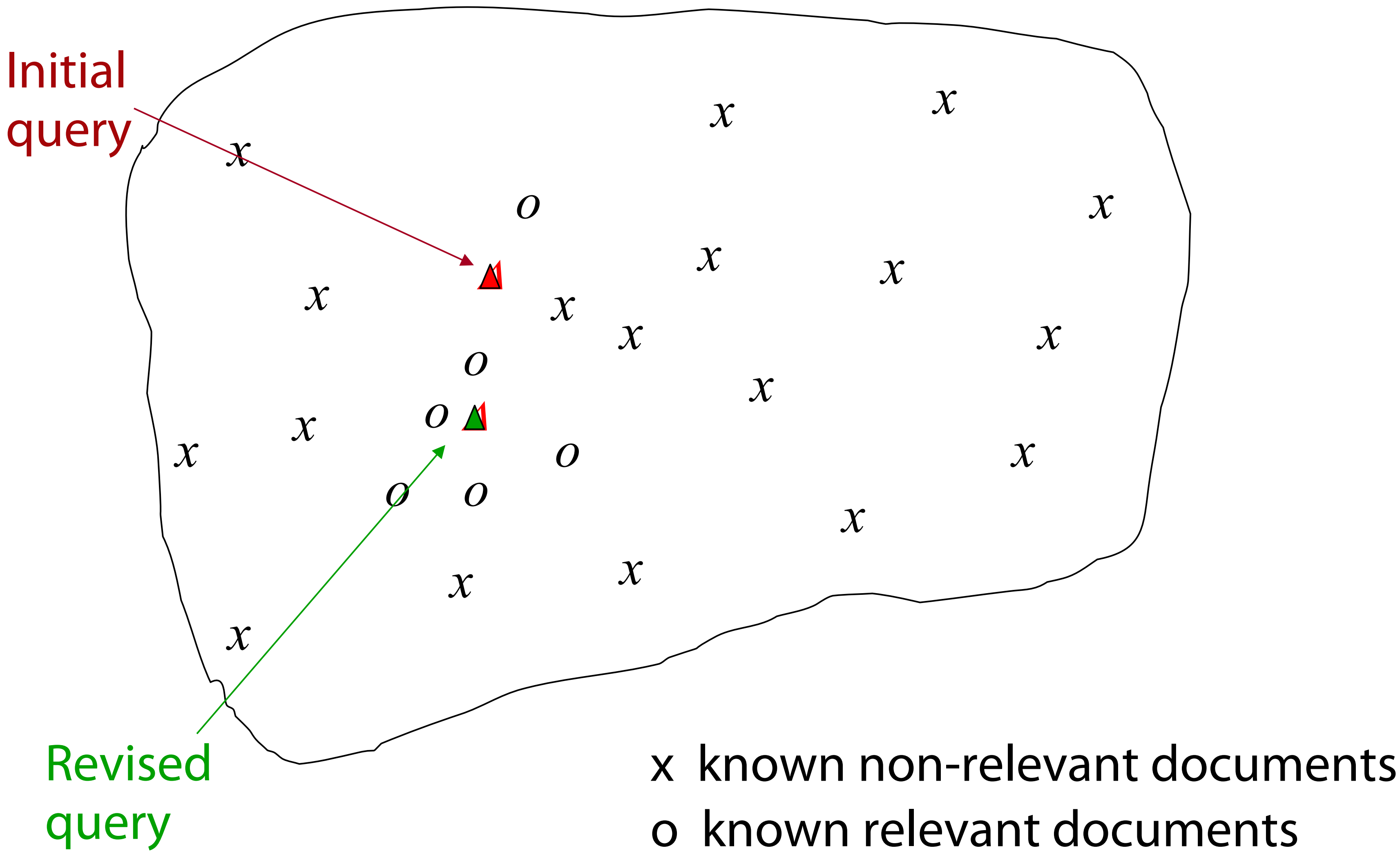
$$\vec{q}_m = \alpha \vec{q}_0 + \beta \frac{1}{|D_r|} \sum_{\vec{d}_j \in D_r} \vec{d}_j - \gamma \frac{1}{|D_{nr}|} \sum_{\vec{d}_j \in D_{nr}} \vec{d}_j$$

- q_m = modified query vector; q_0 = original query vector; α,β,γ: weights (hand-chosen or set empirically); D_r = set of known relevant doc vectors; D_{nr} = set of known irrelevant doc vectors
- New query moves toward relevant documents and away from irrelevant documents
- Tradeoff α vs. β/γ : If we have a lot of judged documents, we want a higher β/γ.
- Term weights ($\vec{q}_m$ elements) can go negative
  - Negative term weights are ignored (set to 0)

Wikipedia: Gerard Salton, The **SMART** (**S**ystem for the **M**echanical **A**nalysis and **R**etrieval of **T**ext or Salton's Magic Automatic Retriever of Text) Information Retrieval System is an information retrieval system, developed at Cornell University in the **1960s**. Many important concepts in information retrieval were developed as part of research on the SMART system, including the vector space model, relevance feedback, and Rocchio algorithm.

Salton, G. (Ed.). *The SMART retrieval system: Experiments in automatic document processing*. Englewood Cliffs, NJ: Prentice-Hall. 1971.

Relevance feedback on initial query



Positive vs Negative Feedback

Positive feedback is more valuable than negative feedback (so, set $\gamma < \beta$; e.g. $\gamma = 0.25$, $\beta = 0.75$).

Why?

Many systems only allow positive feedback ($\gamma=0$).

Relevance feedback in vector spaces

- We can modify the query based on relevance feedback and apply standard vector space model.
- Use only the docs that were marked.
- Relevance feedback can improve recall **and** precision

Any idea for relevance feedback in probabilistic IR?

What about Learning to Rank?

- Embedding data into vector spaces is a very old idea
- Why not using machine learning to find classifiers?
  - Traditional ranking functions in IR used a very small number of features, e.g.,
    - Term frequency
    - Inverse document frequency
    - …
  - Easy to tune weighting coefficients by hand
- More and more features can be defined

Difference between Data Mining and ML?

https://www.discoverdatascience.org/articles/data-mining-vs-machine-learning/

What is the Difference Between Data Mining and Machine Learning?

Data mining is the probing of available datasets in order to identify patterns and anomalies. Machine learning is the process of machines (a.k.a. computers) learning from heterogeneous data in a way that mimics the human learning process. The two concepts together enable both past data characterization and future data prediction.

Well, current ML does not even attempt to show that learning processes are cognitively plausible

In principle, there is no difference between data mining and ML.

Data mining deals with secondary memory (more interesting)

Why is machine learning needed now?

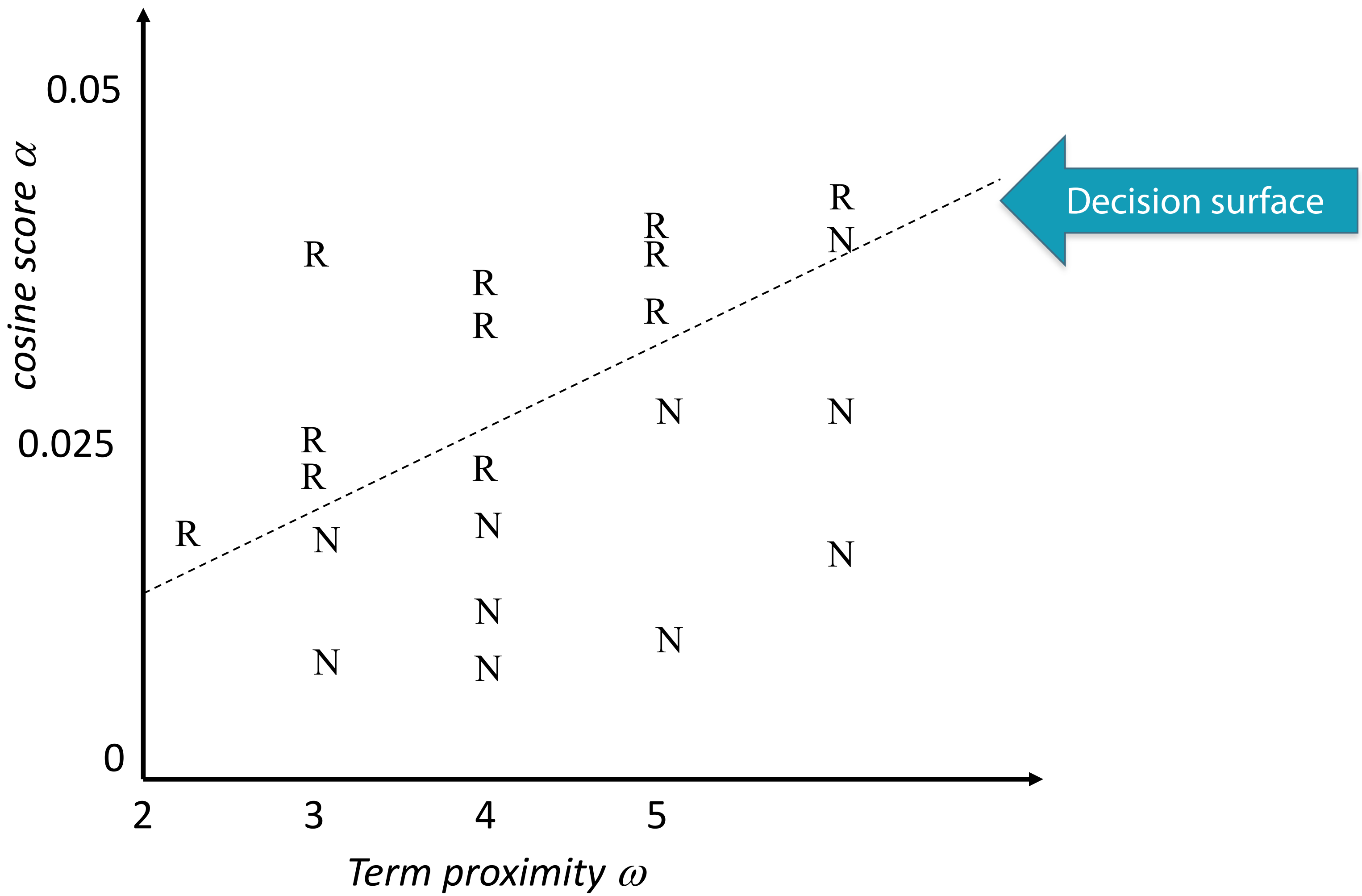
- Modern systems – especially on the Web – use a huge number of features:
  - Log frequency of query word in anchor text?
  - Query word in color on page?
  - # of images on page?
  - # of (out) links on page?
  - PageRank of page? (-> for PageRank, see course EWDS)
  - URL length?
  - URL contains "~"?
  - Page edit recency?
  - Page length?
- *New York Times* (2008-06-03):
  - Google was using over 200 such features
    [in a quotation of a previous Google representative]

Using classification for ad hoc IR

- Collect a training corpus of (*q*, *d*, *r*) triples
  - Relevance *r* is here binary (but may be multiclass, with 3–7 values)
  - Document is represented by a feature vector
    - **x** = (α, ω)   α is cosine similarity, ω is minimum query window size
      - ω is the the shortest text span that includes all query words
      - Query term proximity is a **very important** new weighting factor
  - Train a machine learning model to predict the class *r* of a document-query pair

example	docID	query	cosine score	ω	judgment
ψ_1	37	linux operating system	0.032	3	*relevant*
ψ_2	37	penguin logo	0.02	4	*nonrelevant*
ψ_3	238	operating system	0.043	2	*relevant*
ψ_4	238	runtime environment	0.004	2	*nonrelevant*
ψ_5	1741	kernel layer	0.022	3	*relevant*
ψ_6	2094	device driver	0.03	2	*relevant*
ψ_7	3191	device driver	0.027	5	*nonrelevant*

Using classification for ad hoc IR



Scoring for ad hoc IR

- A linear score function is then

$$Score(d, q) = Score(\alpha, \omega) = a\alpha + b\omega + c$$

Verallgemeinert auf mit Funktion f bezeichnet:

$$f(\psi_i) = w\,\psi_i \text{ with } w = (a, b\ c) \text{ and } \psi_i = (\alpha, \omega, 1)$$

- And the linear classifier is

Decide relevant if $Score(d, q) > \theta$

- … and use score for ranking

Is this a machine learning or a data mining approach?

Support Vector Machines (SVMs → EWDS)

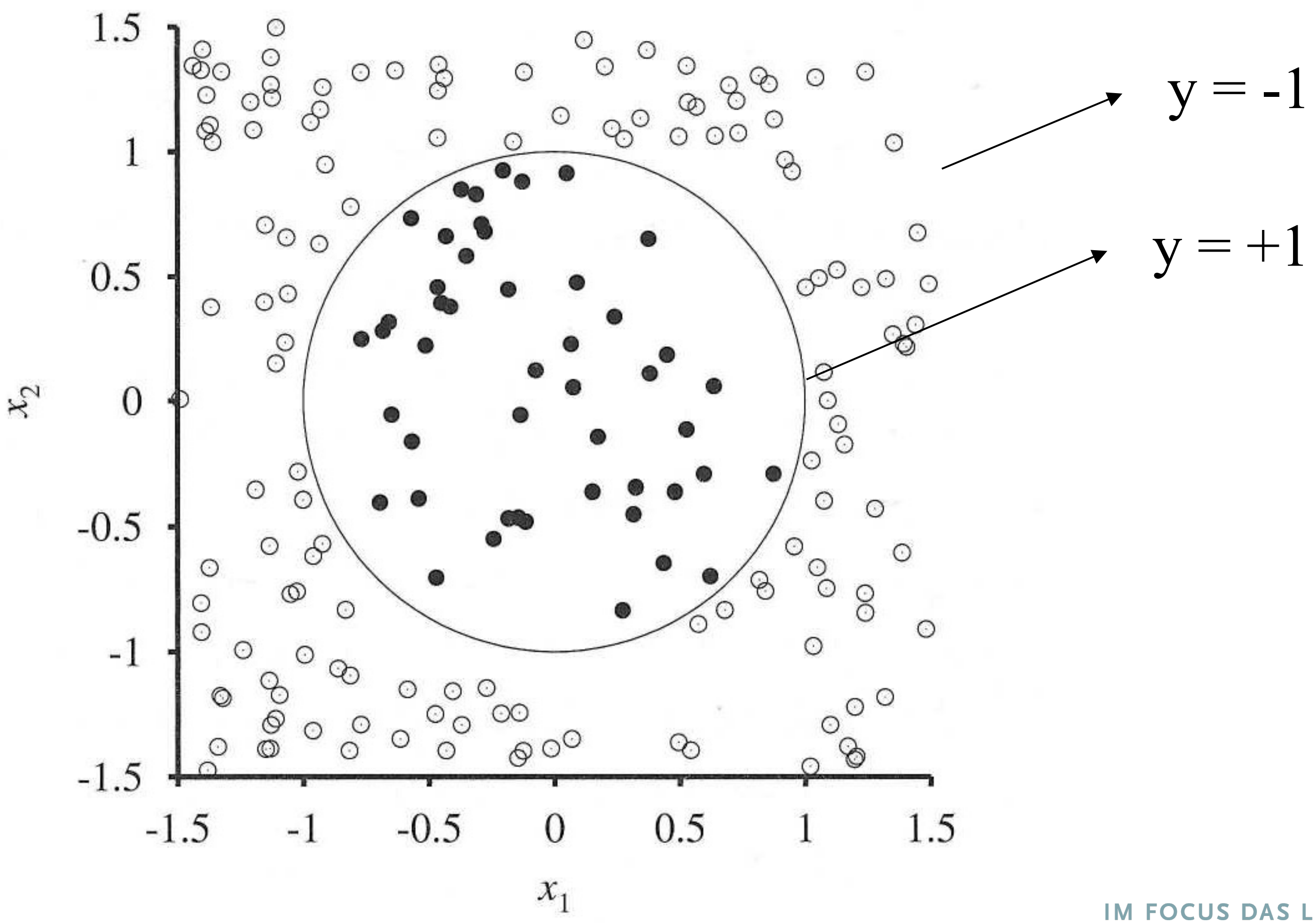
- Mapping instances of two classes into a space, in which they are linearly separable
  - Mapping function is called kernel function
- Computing of separation surface defined via optimization problem
- Formulation as a problem, not as a procedure!

V. Vapnik, A. Chervonenkis, A note on one class of perceptrons. *Automation and Remote Control*, **25**, **1964**

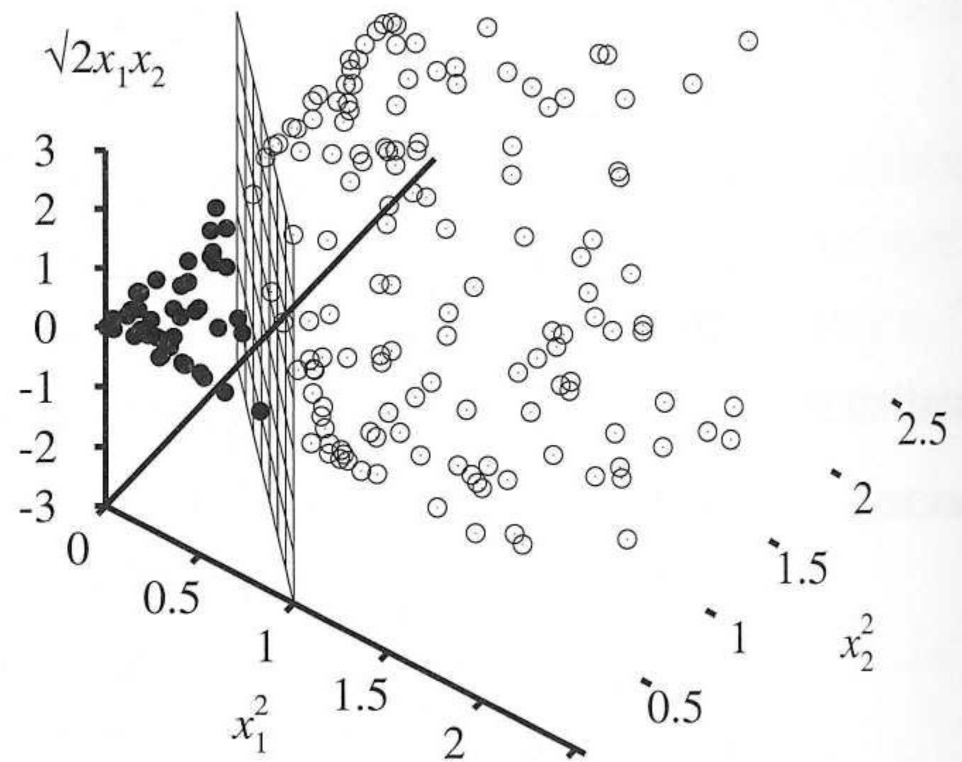
Boser, B. E.; Guyon, I. M.; Vapnik, V. N., A training algorithm for optimal margin classifiers. *Proceedings of the fifth annual workshop on Computational learning theory – COLT '92*. p. 144, **1992**

Vapnik, V., Support-vector networks, Machine Learning. 20 (3): 273–297, **1995**

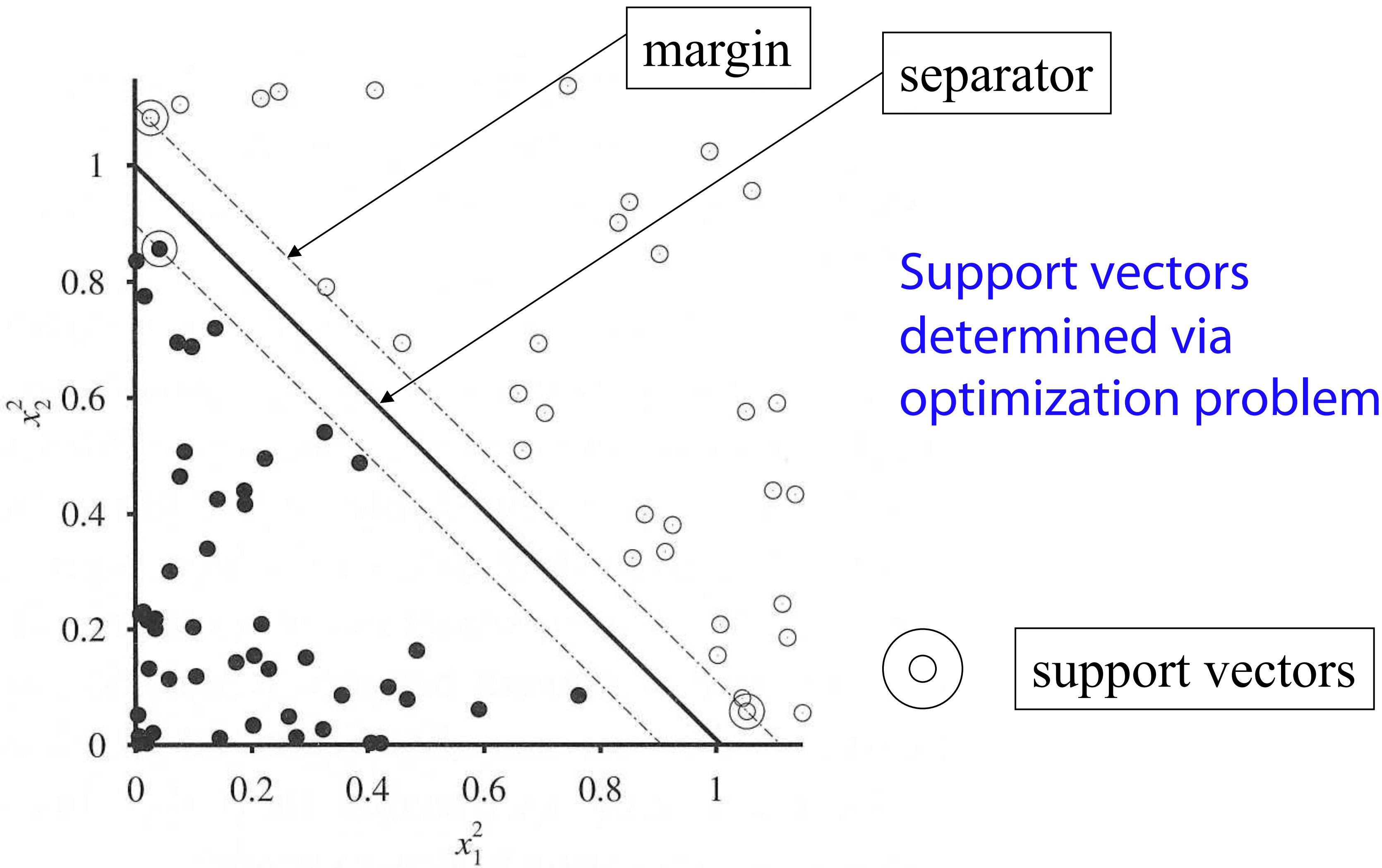
Nonlinear Separation



$(x_1^2, x_2^2, \sqrt{2x_1x_2})$



Support Vectors



Support vectors determined via optimization problem

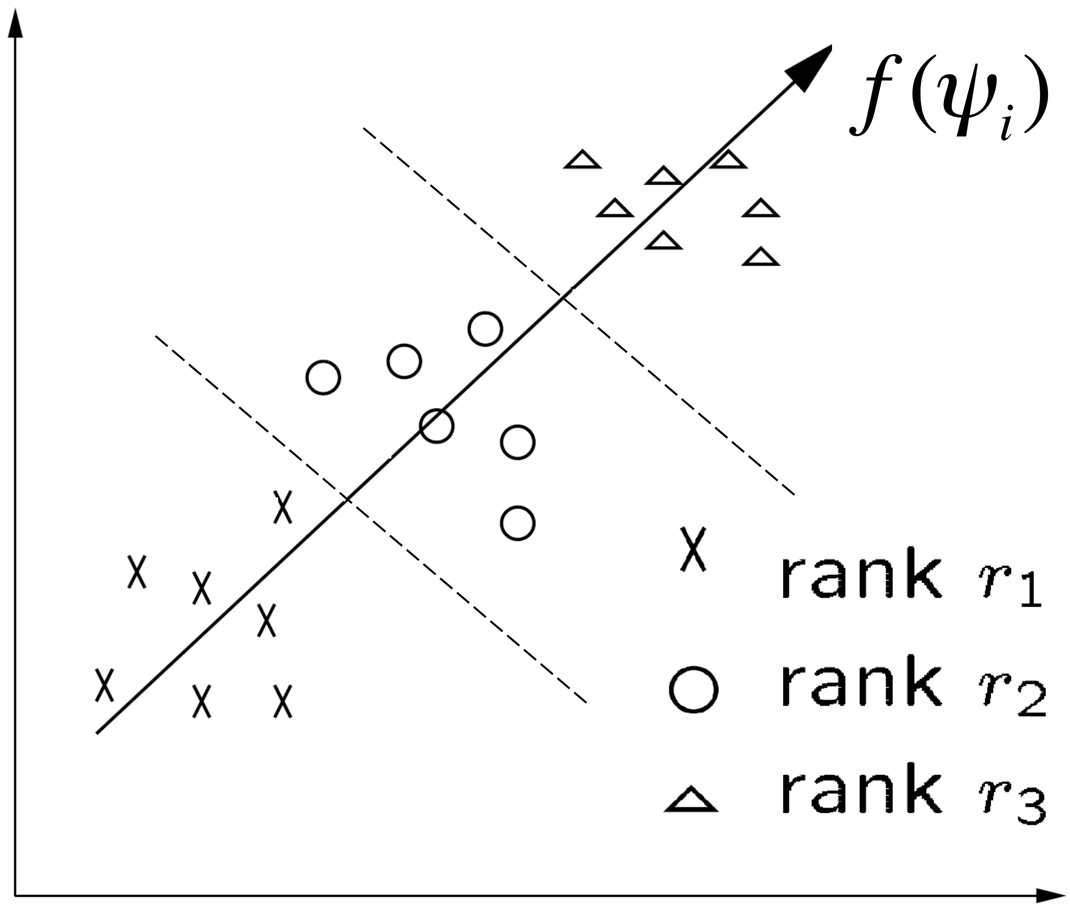
Multi-class SVMs?

- SVMs for multiple class labels
- Combination of multiple SVMs

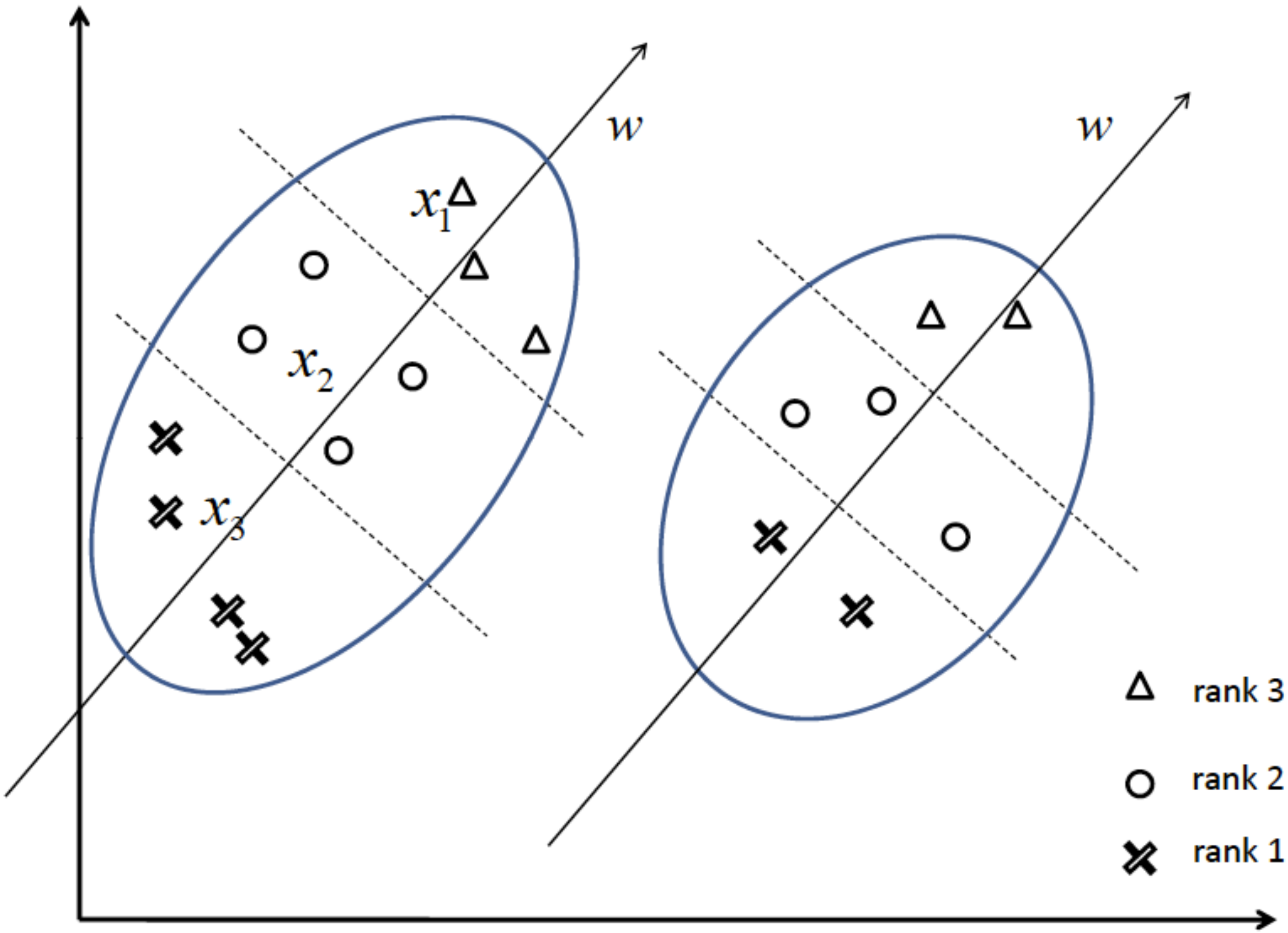
- Assume that classes are ordered (ordinal scale)
- Ordinal regression instead of classification

The Ranking SVM

- Ranking Model: $f(\psi_i)$, e. g. $f(\psi_i) = w\,\psi_i$ (linear model)



Two queries in the original space



An SVM classifier for information retrieval

- Experiments:
  - 4 TREC data sets
    (Data3, Data4, Data5, WT10G (web))
  - Comparisons with Lemur (LM),
    a state-of-the-art open source IR engine
  - Linear kernel normally best or almost as good as quadratic kernel, and so used in reported results
  - 6 features, all variants of tf, idf, and tf.idf scores

Ramesh Nallapati. Discriminative models for information retrieval. In Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '04). Association for Computing Machinery, New York, NY, USA, 64–71. **2004**.

An SVM classifier for information retrieval

Train \ Test		Disk 3	Disk 4-5	WT10G (web)
Disk 3	LM	**0.1785**	**0.2503**	0.2666
	SVM	0.1728	0.2432	**0.2750**
Disk 4-5	LM	**0.1773**	**0.2516**	0.2656
	SVM	0.1646	0.2355	**0.2675**

- At best the results are about equal to LM
  - Actually, a little bit below
- Paper's advertisement: Easy to add more features
  - This is illustrated on a homepage finding task on WT10G:
  - Baseline LM 52% success@10, baseline SVM 58%
  - SVM with URL-depth, and in-link features: 78% S@10

Overview

- Moderate number of features
  - SVMs: 2000-2007
- Very high number of features
  - Deep composition of high-dimensional linear and piecewise linear functions: 2007-2014

    + Learn latent features in different composition layers

    → EWDS
- Input sequences (order between dimensions)
  - Transformer networks: 2014 …

    + Learn influence weights of different parts of input ("attention")

IR Agents: Summary

- Goal: Fulfill information need of human user
  - Information need specified in various ways (e.g., query vector)
  - Agent employs strategies to best fulfill its goal(s)
- Agent receives reinforcement feedback ("reward") (e.g., as relevance feedback)
- Agent changes its goal fulfillment strategies for dealing with the same or similar goals
  - E.g., by applying the Rocchio Algorithm
- Agent possibly extends its model of the user
- Agent could refine goals to meet expectations
  - Reduce uncertainty
- Agent could contact other agents to acquire new information