12.11.2024

# Statistical Methods in AI (CS7.403)

## Lecture-25: CV, Retrieval, Metrics, PR,ROC

Ravi Kiran (ravi.kiran@iiit.ac.in)

https://ravika.github.io

@vikataravi

Center for Visual Information Technology (CVIT)
IIIT Hyderabad

# Evaluation

- Evaluation = Process of judging the merit or worth of something
- Evaluation is key to building *effective* and *efficient* Data Science systems
  - usually carried out in controlled experiments
  - *online* testing can also be done

# Why System Evaluation?

- There are many models/ algorithms/ systems, which one is the best?
- What is the best component for:
  - similarity function (cosine, correlation,…)
  - Term selection (stopword removal, stemming…)
  - Term weighting (TF, TF-IDF,…)
- How far down the list will a user need to look to find some/all relevant documents in text retrieval?

# Regression / classification models

- Predictive modeling / Supervised learning
- A model is a specification of mathematical/probabilistic relationships that exist between different variables
- The goal is usually to use existing data to develop models that we can use to predict outcomes for new data, such as
  - Predicting whether an email message is spam or not
  - Predicting whether a credit card transaction is fraudulent
  - Predicting which advertisement a shopper is most likely to click on
  - Predicting which football team is going to win the Super Bowl

Nominal (categorical with No particular order)

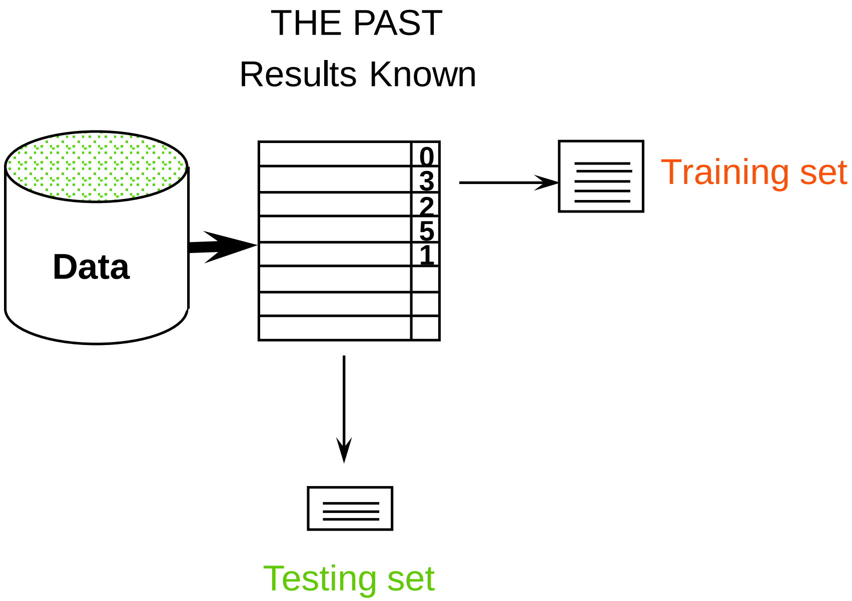
  - Predicting stock price of a given company
  - Predicting number of buyers of a certain product
  - Predicting user ratings of a new movie
  - Predicting the grade of a disease

Continuous / ordinal

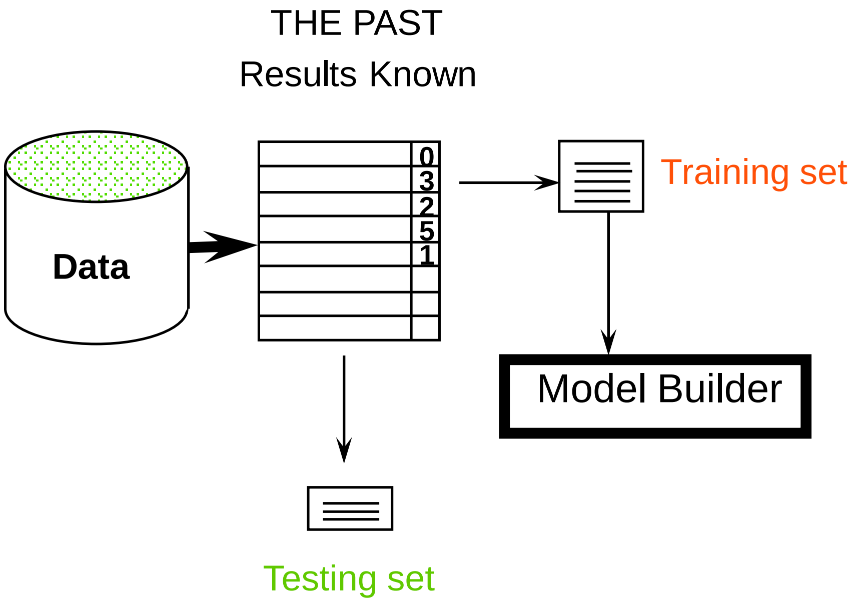
# Evaluation on "LARGE" data

- If many (thousands) of examples are available, then how can we evaluate our model?
- A simple evaluation is sufficient
  - Randomly split data into training and test sets (e.g. 2/3 for train, 1/3 for test)
  - For classification, make sure training and testing have similar distribution of class labels
- Build a model using the *train* set and evaluate it using the *test* set.

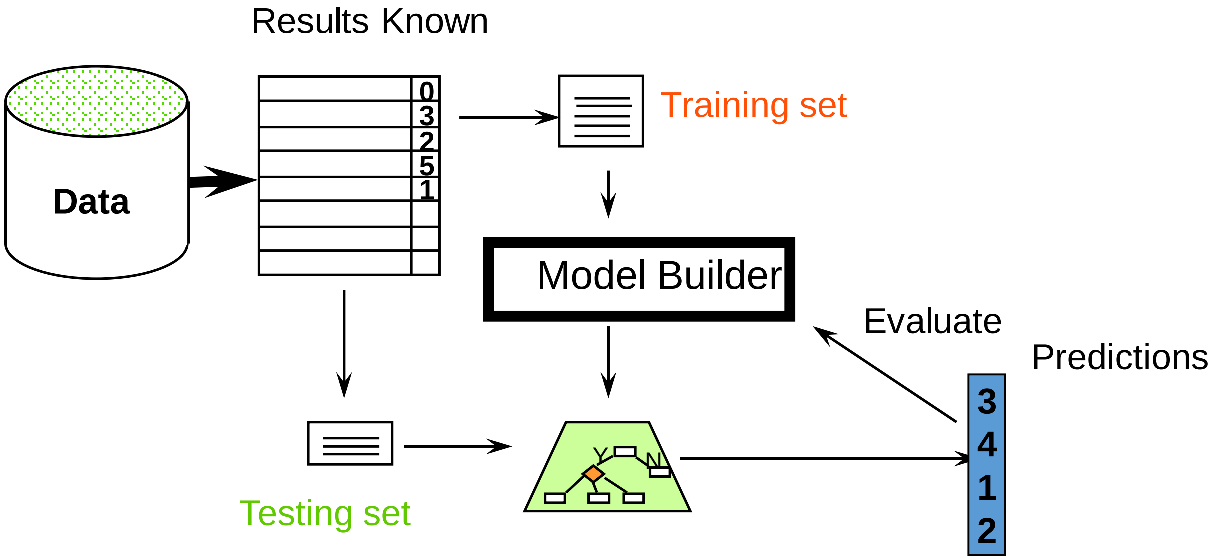
# Model Evaluation Step 1:
# Split data into train and test sets



# Model Evaluation Step 2:
# Build a model on a training set



# Model Evaluation Step 3:
# Evaluate on test set



# A note on parameter tuning

- It is important that the test data is not used *in any way* to build the model
- Some learning schemes operate in two stages:
  - Stage 1: builds the basic structure
  - Stage 2: optimizes parameter settings
- The test data can’t be used for parameter tuning!
- Proper procedure uses three sets: **training data, validation data, and test data**
  - Validation data is used to optimize parameters

# Evaluation on “small” data, 1

- The *holdout* method reserves a certain amount for testing and uses the remainder for training
  - Usually: one third for testing, the rest for training
- For “unbalanced” datasets, samples might not be representative
  - Few or none instances of some classes
- *Stratified sample: advanced version of balancing  the data*
  - Make sure that each class is represented with approximately equal proportions in both subsets

# Evaluation on “small” data, 2

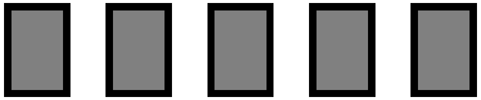
- What if we have a small data set?
- The chosen 2/3 for training may not be representative.
- The chosen 1/3 for testing may not be representative.

# Cross-validation

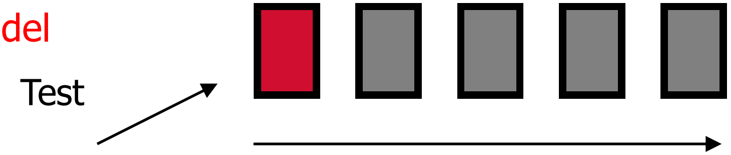
- *Cross-validation more useful in small datasets*
  - First step: data is split into *k* subsets of equal size
  - Second step: each subset in turn is used for testing and the remainder for training
- This is called *k-fold cross-validation*
- For classification, often the subsets are stratified before the cross-validation is performed
- The error estimates are averaged to yield an overall error estimate

# Cross-validation example:

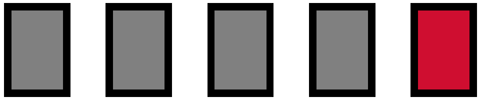
- Break up data into groups of the same size



- Hold aside one group for testing and use the rest to build model



- Repeat



# More on cross-validation

- Standard method for evaluation: stratified ten-fold cross-validation
- Why ten? Extensive experiments have shown that this is the best choice to get an accurate estimate
- Stratification reduces the estimate’s variance
- Even better: repeated stratified cross-validation
  - E.g. ten-fold cross-validation is repeated ten times and results are averaged (reduces the variance)

# Confusion Matrix

A confusion matrix is a table that is often used to **describe the performance of a classification model** (or "classifier") on a set of test data

- true positives (TP): These are cases in which we predicted positive (they have the disease), and they do have the disease.
- true negatives (TN): We predicted negative, and they don't have the disease.
- false positives (FP): We predicted positive, but they don't actually have the disease. (Also known as a "Type I error.")
- false negatives (FN): We predicted negative, but they actually do have the disease. (Also known as a "Type II error.")

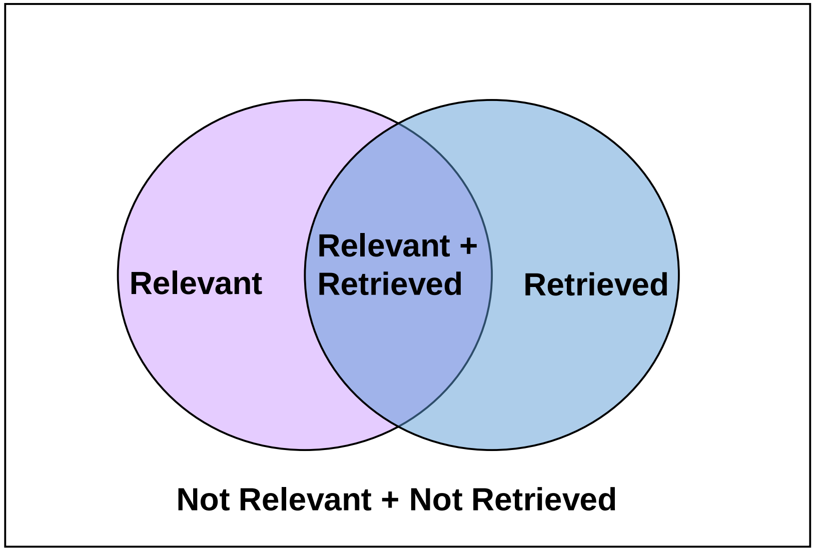
| | Actual: Positive | Actual: Negative |
|---|---|---|
| Predicted: Positive | tp | fp |
| Predicted: Negative | fn | tn |

# Precision and Recall in Text Retrieval

- Precision
  - The ability to retrieve top-ranked documents that are mostly relevant.
  - Precision P = tp/(tp + fp)
- Recall
  - The ability of the search to find ***all*** of the relevant items in the corpus.
  - Recall      R = tp/(tp + fn)

| | Relevant | Nonrelevant |
|---|---|---|
| Retrieved | tp | fp |
| Not Retrieved | fn | tn |

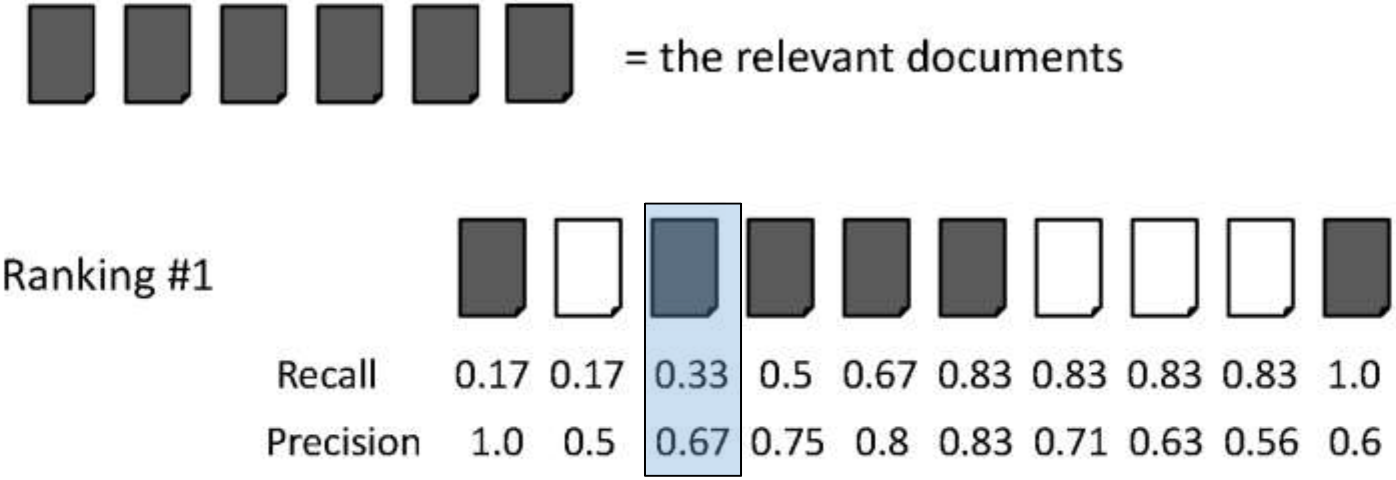
# Precision and Recall : Retrieval



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

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

# Precision/Recall : Example


= 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 |

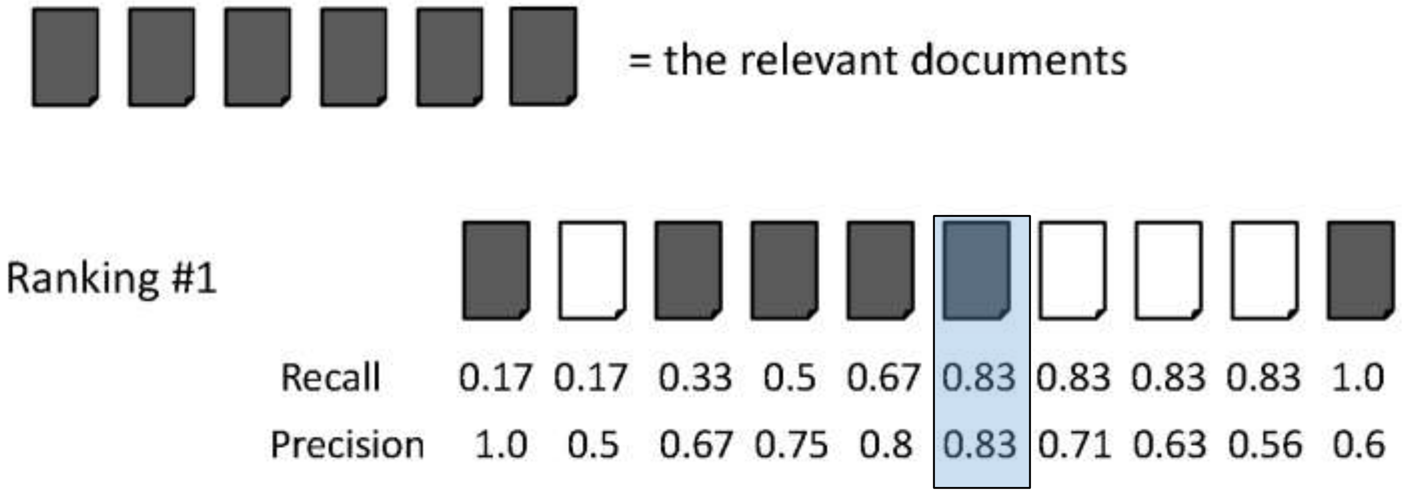
Recall = 2/6 = 0.33

Precision = 2/3 = 0.67

$$\textbf{Precision} = \frac{\textbf{Relevant Retrieved}}{\textbf{Retrieved}}$$

$$\textbf{Recall} = \frac{\textbf{Relevant Retrieved}}{\textbf{Relevant}}$$

# Precision/Recall : Example



| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 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 |

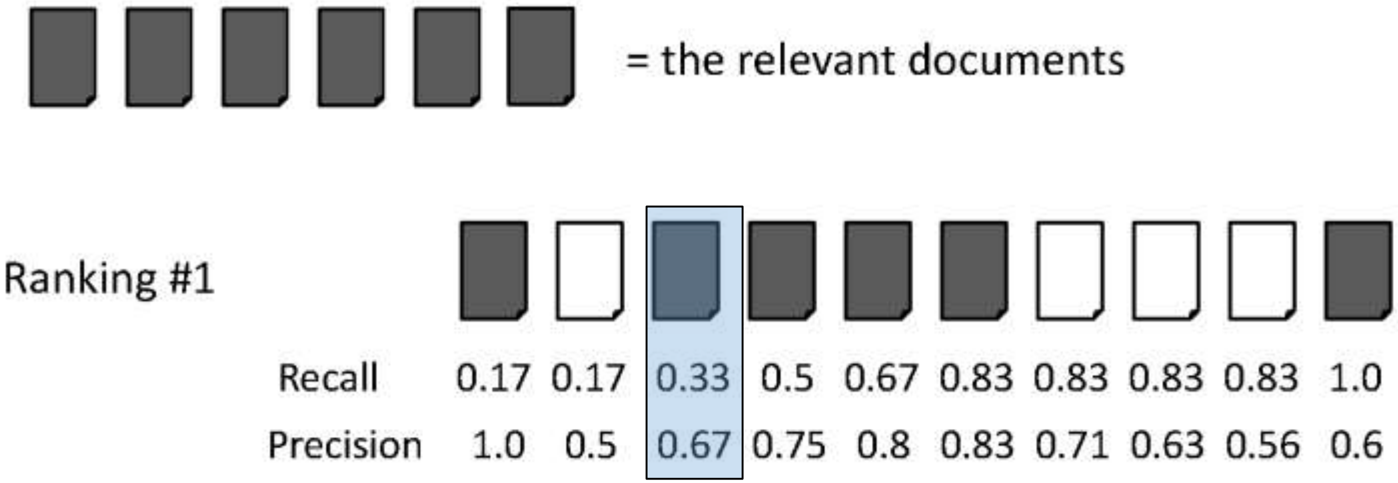
Recall = 5/6 = 0.83

Precision = 5/6 = 0.83

$$\textbf{Precision} = \frac{\textbf{Relevant Retrieved}}{\textbf{Retrieved}}$$

$$\textbf{Recall} = \frac{\textbf{Relevant Retrieved}}{\textbf{Relevant}}$$

# F Measure (F1/Harmonic Mean) : example



|  |  |  |  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|
| 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 |

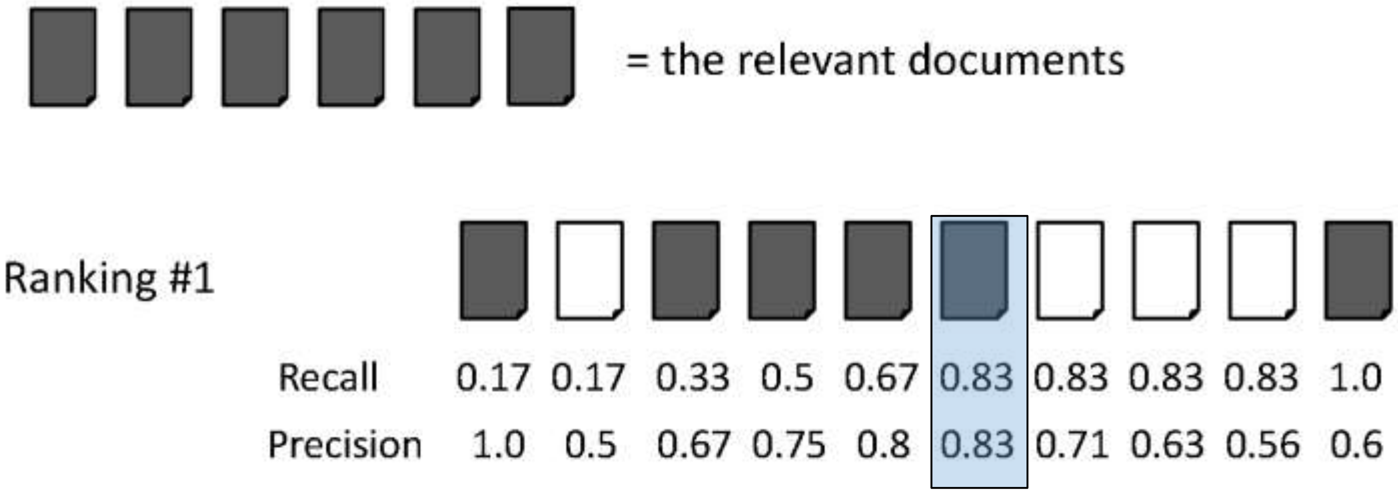
Recall = 2/6 = 0.33

Precision = 2/3 = 0.67

F = 2*Recall*Precision/(Recall + Precision)

= 2*0.33*0.67/(0.33 + 0.67) = 0.44

# F Measure (F1/Harmonic Mean) : example



| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 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 |

Recall = 5/6 = 0.83

Precision = 5/6 = 0.83

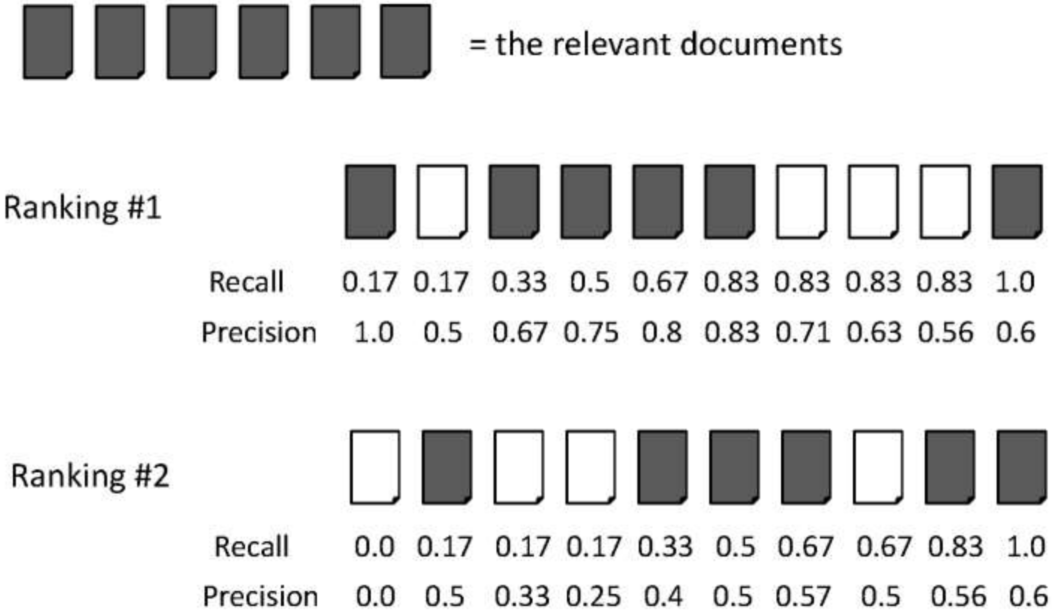
F = 2*Recall*Precision/(Recall + Precision)

= 2*0.83*0.83/(0.83 + 0.83) = 0.83

# Mean Average Precision (MAP)

- **Average Precision**: Average of the precision values at the points at which each relevant document is retrieved.
  - Ex1: (1 + 1 + 0.75 + 0.667 + 0.38 + 0)/6 = 0.633
  - Ex2: (1 + 0.667 + 0.6 + 0.5 + 0.556 + 0.429)/6 = 0.625
  - Averaging the precision values from the rank positions where a relevant document was retrieved
  - Set precision values to be zero for the not retrieved documents

# Average Precision: Example



■ ■ ■ ■ ■ ■ = the relevant documents

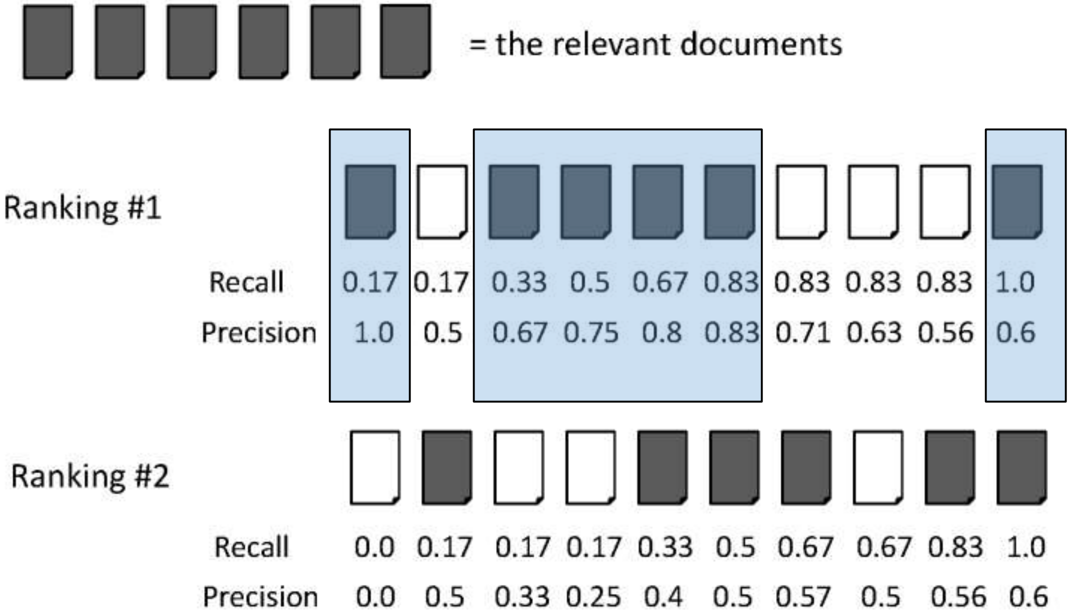
Ranking #1

|  |  |  |  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|
|  | ■ | □ | ■ | ■ | ■ | ■ | □ | □ | □ | ■ |
| 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 |

# Average Precision: Example



= the relevant documents

Ranking #1

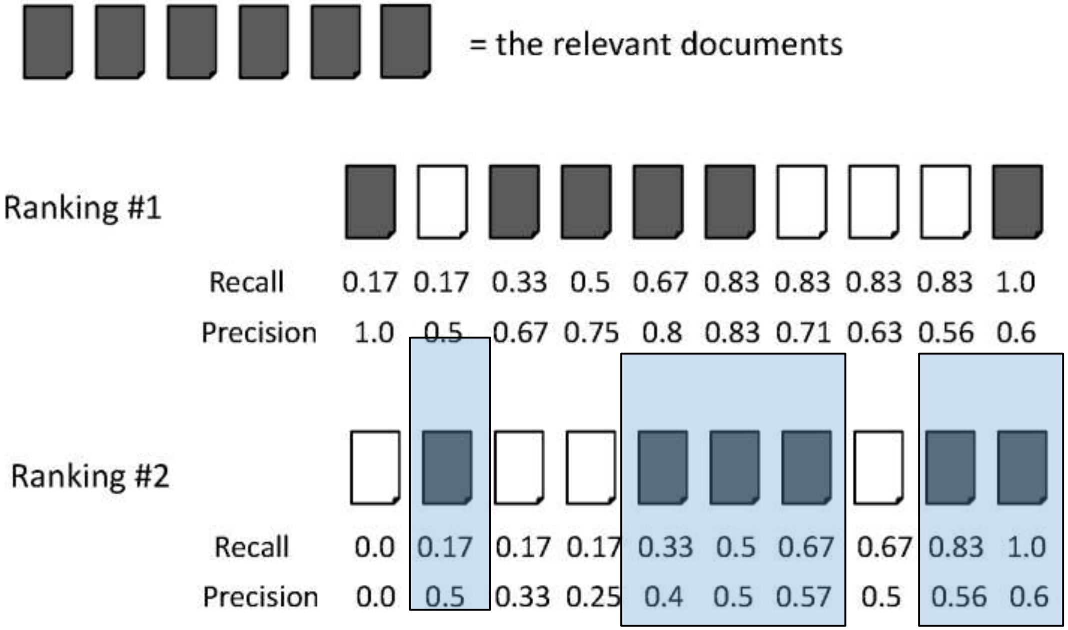
| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Recall | 0.17 | 0.17 | 0.33 | 0.5 | 0.67 | 0.83 | 0.83 | 0.83 | 0.83 | 1.0 |
| Precision | 1.0 | 0.5 | 0.67 | 0.75 | 0.8 | 0.83 | 0.71 | 0.63 | 0.56 | 0.6 |

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 |

$$\text{Ranking \#1:}\ (1.0 + 0.67 + 0.75 + 0.8 + 0.83 + 0.6)/6 = 0.78$$

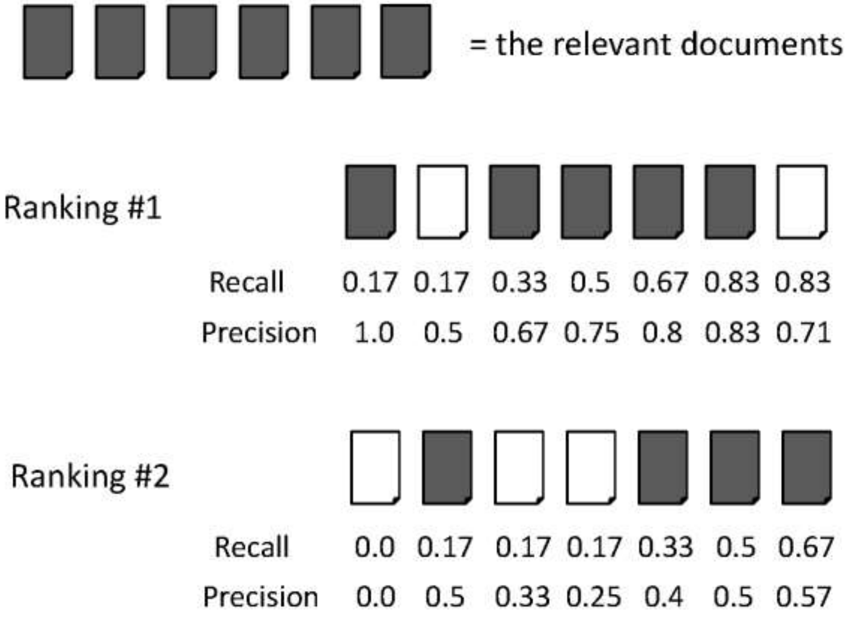
# Average Precision: Example


= 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

$$\text{Ranking \#1: } (1.0 + 0.67 + 0.75 + 0.8 + 0.83 + 0.6)/6 = 0.78$$

$$\text{Ranking \#2: } (0.5 + 0.4 + 0.5 + 0.57 + 0.56 + 0.6)/6 = 0.52$$

# Average Precision: Example



Ranking #1

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Recall | 0.17 | 0.17 | 0.33 | 0.5 | 0.67 | 0.83 | 0.83 |
| Precision | 1.0 | 0.5 | 0.67 | 0.75 | 0.8 | 0.83 | 0.71 |

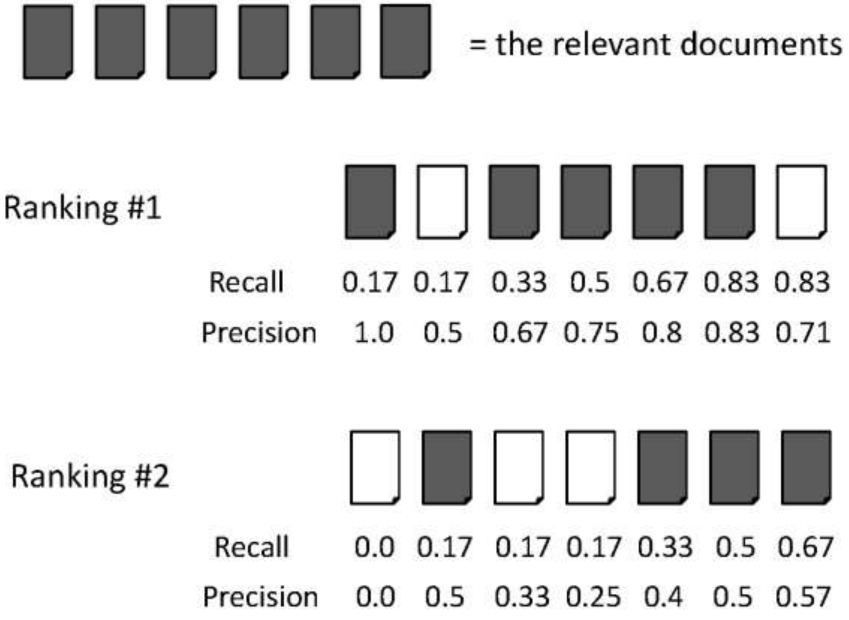
Ranking #2

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Recall | 0.0 | 0.17 | 0.17 | 0.17 | 0.33 | 0.5 | 0.67 |
| Precision | 0.0 | 0.5 | 0.33 | 0.25 | 0.4 | 0.5 | 0.57 |

Miss one relevant document

$$\begin{aligned}\text{Rank 1} &= (1 + 0.67 + 0.75 + 0.8 + 0.83 + \boxed{0})/6 = 0.675 \\ \text{Rank 2} &= (0.5 + 0.4 + 0.5 + 0.57 + 0 + 0)/6 = 0.328\end{aligned}$$

# Average Precision: Example



= the relevant documents

Ranking #1

| Recall | 0.17 | 0.17 | 0.33 | 0.5 | 0.67 | 0.83 | 0.83 |
|---|---|---|---|---|---|---|---|
| Precision | 1.0 | 0.5 | 0.67 | 0.75 | 0.8 | 0.83 | 0.71 |

Ranking #2

| Recall | 0.0 | 0.17 | 0.17 | 0.17 | 0.33 | 0.5 | 0.67 |
|---|---|---|---|---|---|---|---|
| Precision | 0.0 | 0.5 | 0.33 | 0.25 | 0.4 | 0.5 | 0.57 |

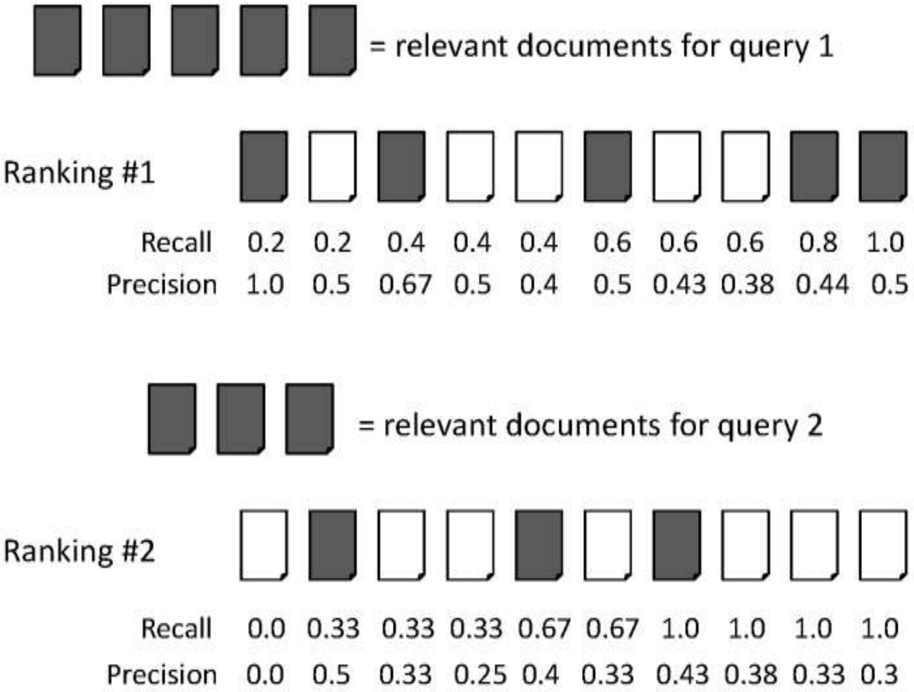
Miss two relevant documents

$$\begin{aligned}\text{Rank 1} &= (1 + 0.67 + 0.75 + 0.8 + 0.83 + 0)/6 = 0.675 \\ \text{Rank 2} &= (0.5 + 0.4 + 0.5 + 0.57 + \boxed{0 + 0})/6 = 0.328\end{aligned}$$

# Mean Average Precision (MAP)

- Summarize rankings from multiple queries by averaging average precision
- Most commonly used measure in research papers
- Assumes user is interested in finding **many** relevant documents for each query
- Requires **many** relevance judgments in text collection

# Mean Average Precision (MAP)


= relevant documents for query 1
Ranking #1
= relevant documents for query 2
Ranking #2

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 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 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 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 |

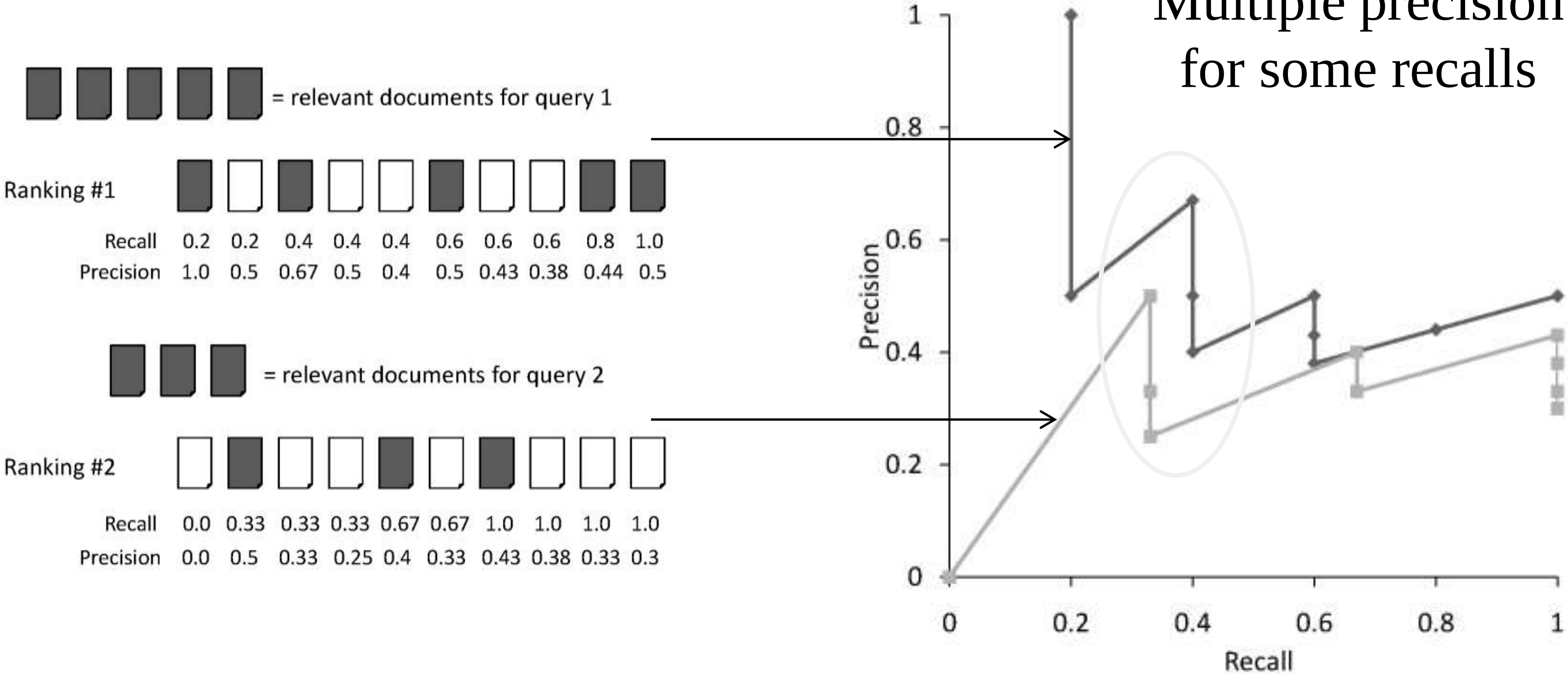
$$\textit{average precision query 1} = (1.0 + 0.67 + 0.5 + 0.44 + 0.5)/5 = 0.62$$
$$\textit{average precision query 2} = (0.5 + 0.4 + 0.43)/3 = 0.44$$

$$\textit{mean average precision} = (0.62 + 0.44)/2 = 0.53$$

# Recall-Precision Graph

- The Recall-Precision Graph is created using the standard Recall values from the Recall Level and Precision Averages.
- Typically these graphs slope downward from left to right, enforcing the notion that as more relevant documents are retrieved (recall increases), the more nonrelevant documents are retrieved (precision decreases).
- This graph is the most commonly used method for comparing systems. The plots of different runs can be superimposed on the same graph to determine which run is superior.
- Curves closest to the upper right-hand corner of the graph (where recall and precision are maximized) indicate the best performance

# Recall-Precision Graph



Multiple precision for some recalls

= 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 |

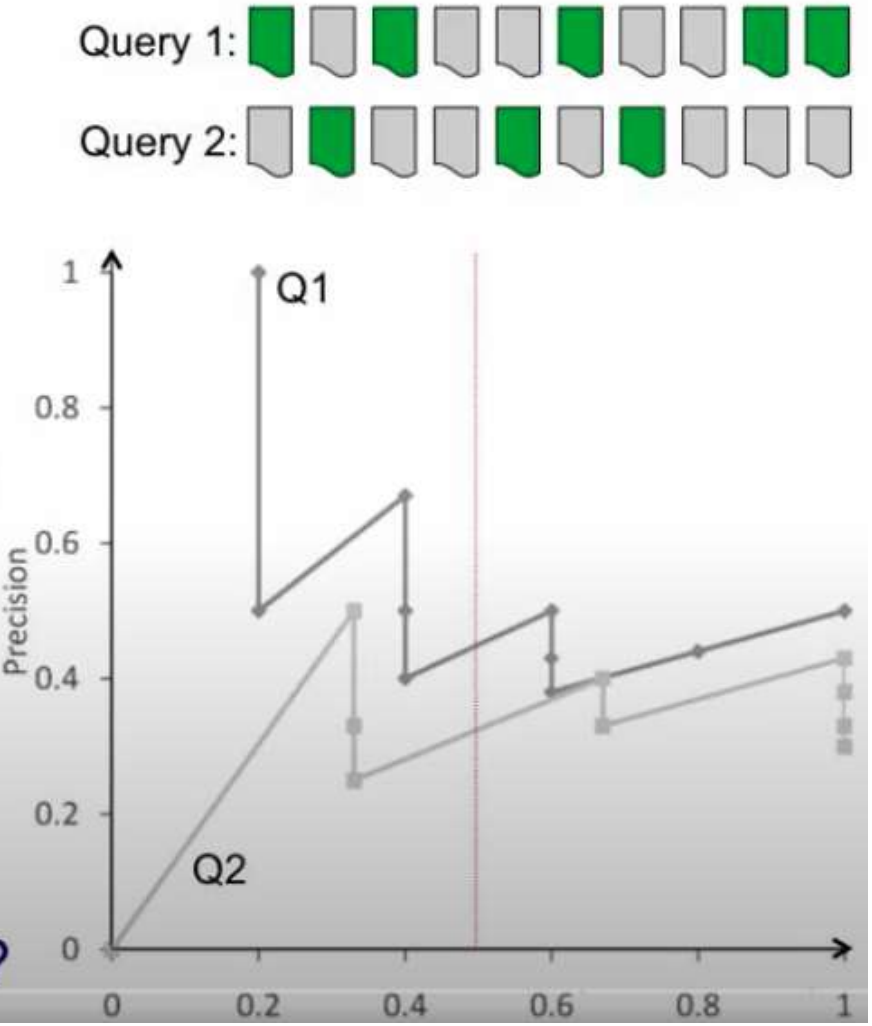
= relevant documents for query 2

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 |

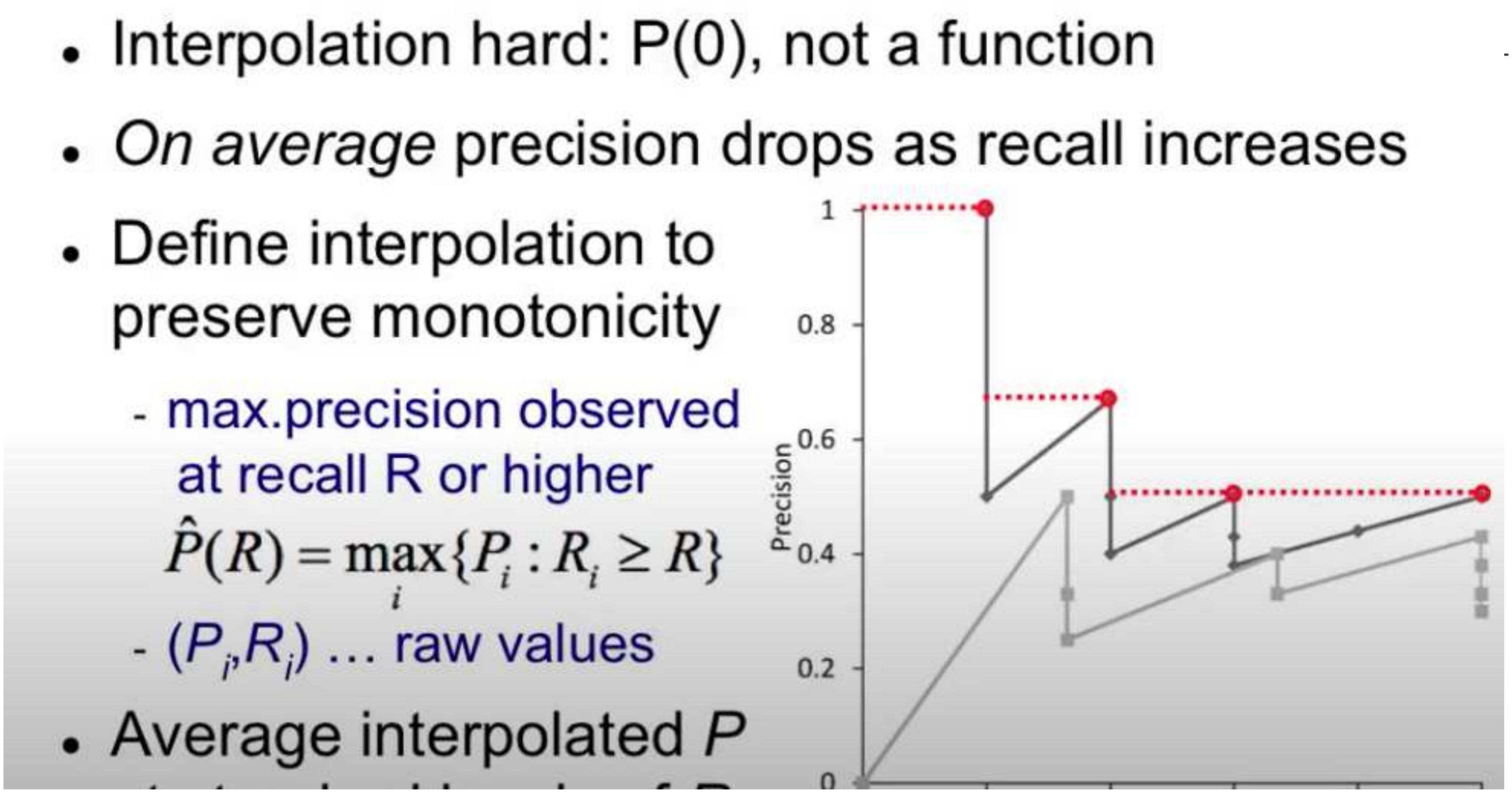
# Recall-precision plot (raw)

- Plot precision vs. recall
  - one curve per query
  - detailed picture, but...
    - erratic behaviour
    - want to “average” curves
- Standard averaging
  - at fixed recall levels
    - 0.1, 0.2, 0.3 … 1.0
  - what is precision at 0.5?
  - need to interpolate, how?



# Interpolation

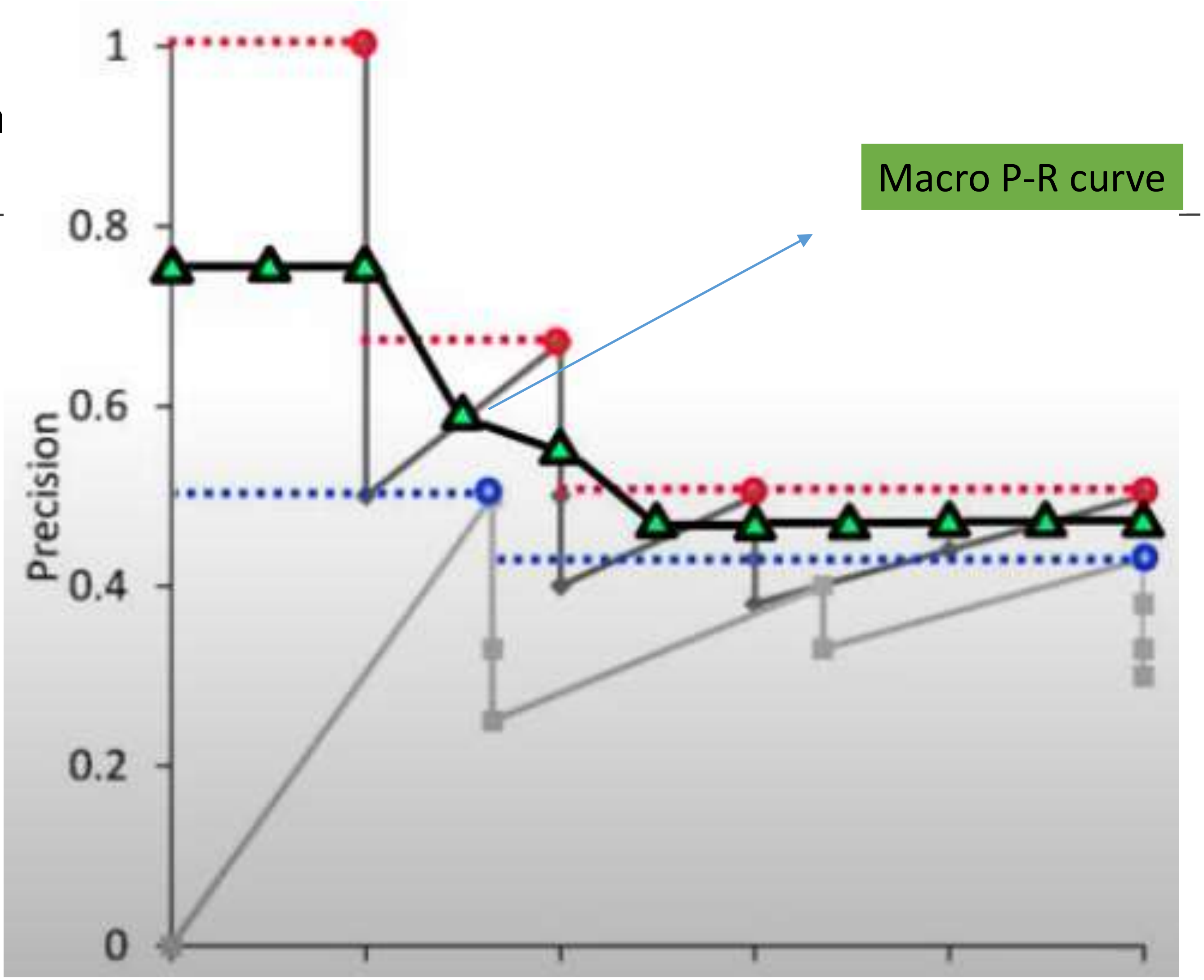
- Defines precision at any recall level as the *maximum* precision observed in any recall-precision point at a higher recall level
  - produces a step function
  - defines precision at recall 0.0, 0.1……1.0



- Interpolation hard: P(0), not a function
- *On average* precision drops as recall increases
- Define interpolation to preserve monotonicity
  - max.precision observed at recall R or higher
    $$\hat{P}(R) = \max_i\{P_i : R_i \geq R\}$$
  - $(P_i, R_i)$ … raw values
- Average interpolated *P* at standard levels of R

Recall

# Average Precision

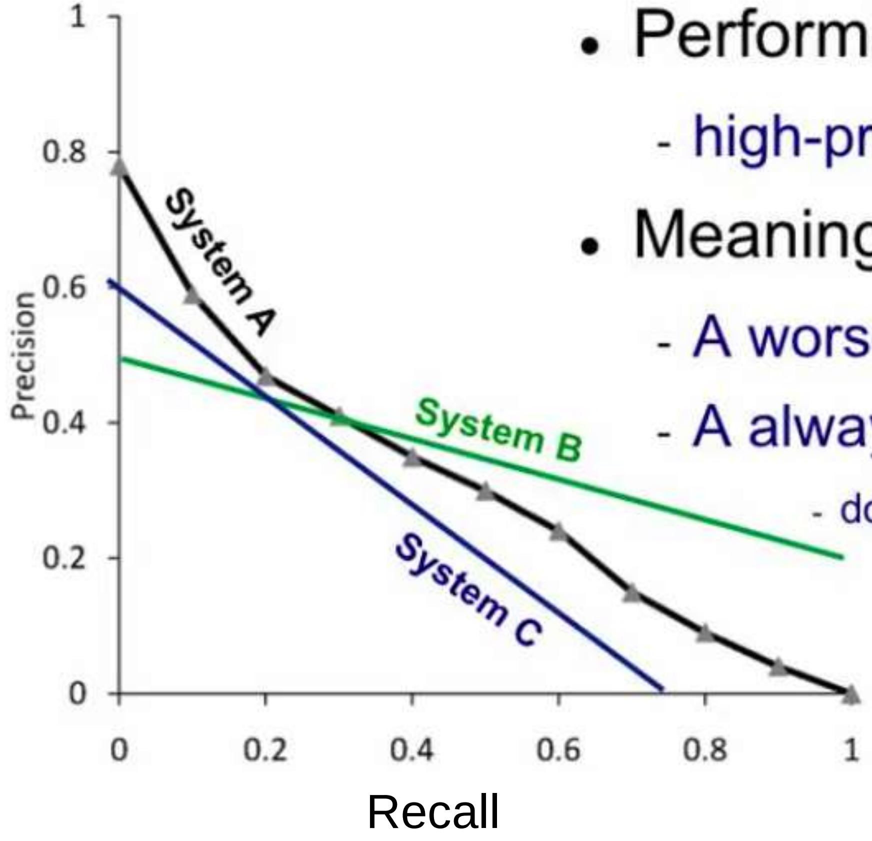


Recall

Average Precision (AP) = mean_q AP_q (Area under P-R curve of query q)

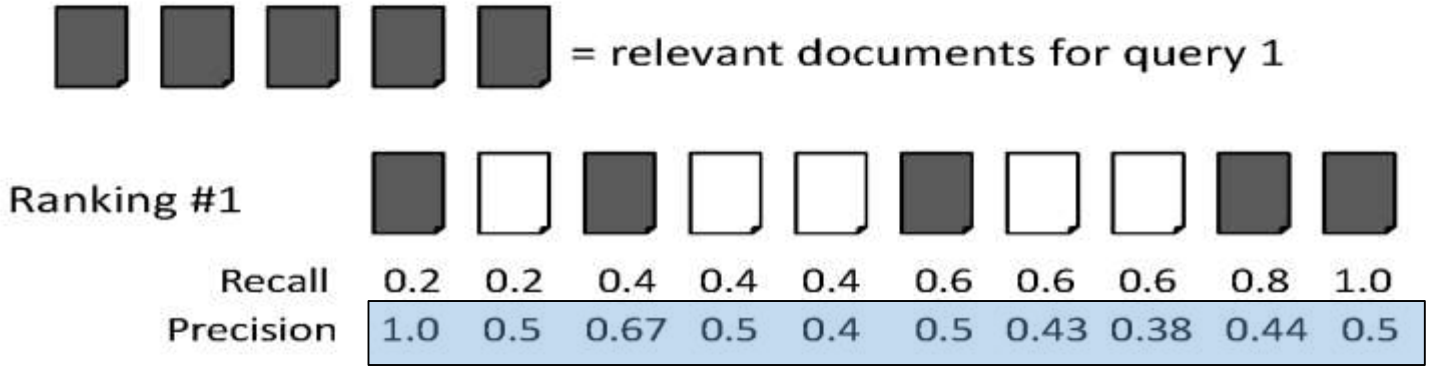
Mean Average Precision (mAP) = mean_c AP[c]

# Comparing retrieval systems using the “average” P-R curves



- Averaged over 50 queries
- Performance for all user types
  - high-precision and high-recall
- Meaningful system comparisons
  - A worse than B if recall important
  - A always better than C
    - dominates at all recall levels

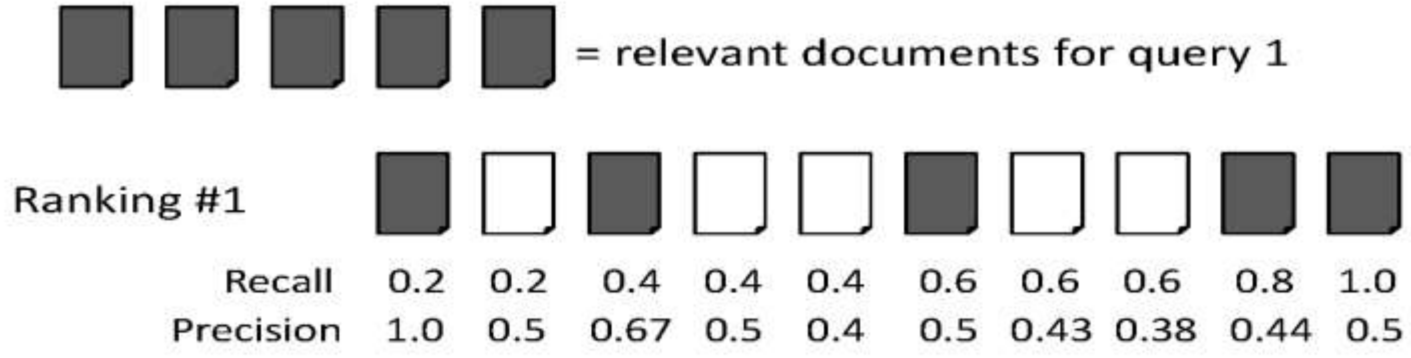
# Interpolation



| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 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 |

| Recall | 0.0 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 | 1.0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Interpolated Precision | 1.0 | | | | | | | | | | |

# Interpolation



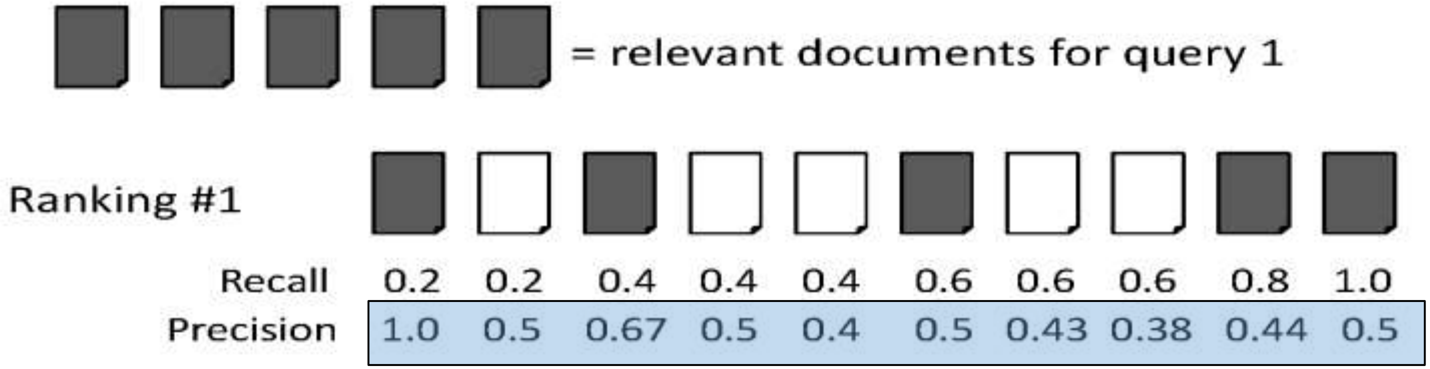
= relevant documents for query 1

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 |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Recall | 0.0 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 | 1.0 |
| Interpolated Precision | 1.0 | | | | | | | | | | |

# Interpolation



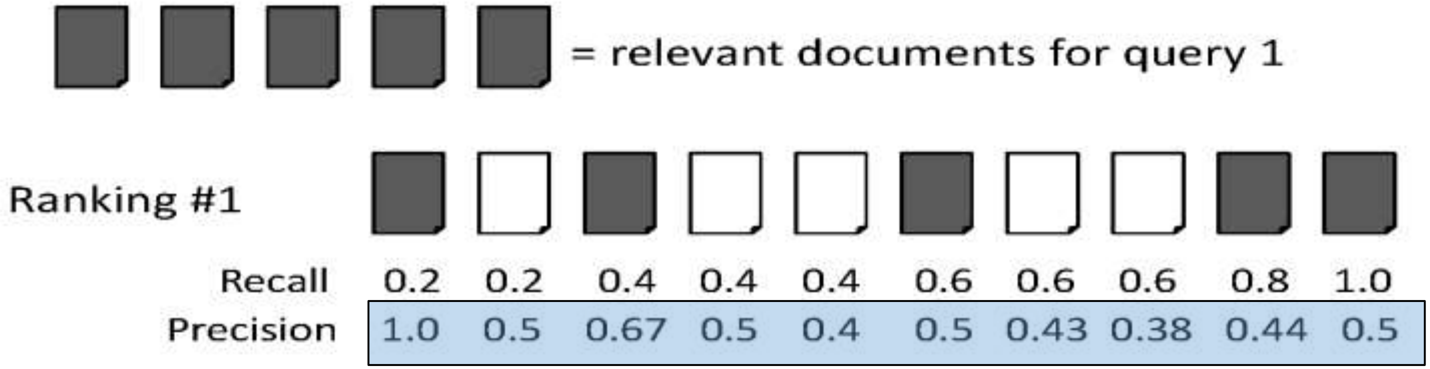
= relevant documents for query 1

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 |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Recall | 0.0 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 | 1.0 |
| Interpolated Precision | 1.0 | 1.0 | | | | | | | | | |

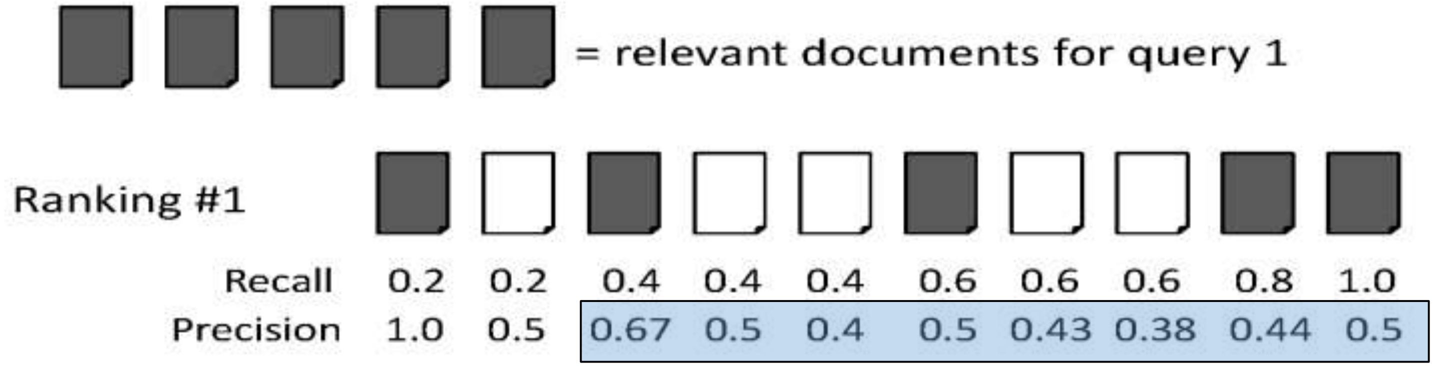
# Interpolation



| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 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 |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Recall | 0.0 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 | 1.0 |
| Interpolated Precision | 1.0 | 1.0 | 1.0 | | | | | | | | |

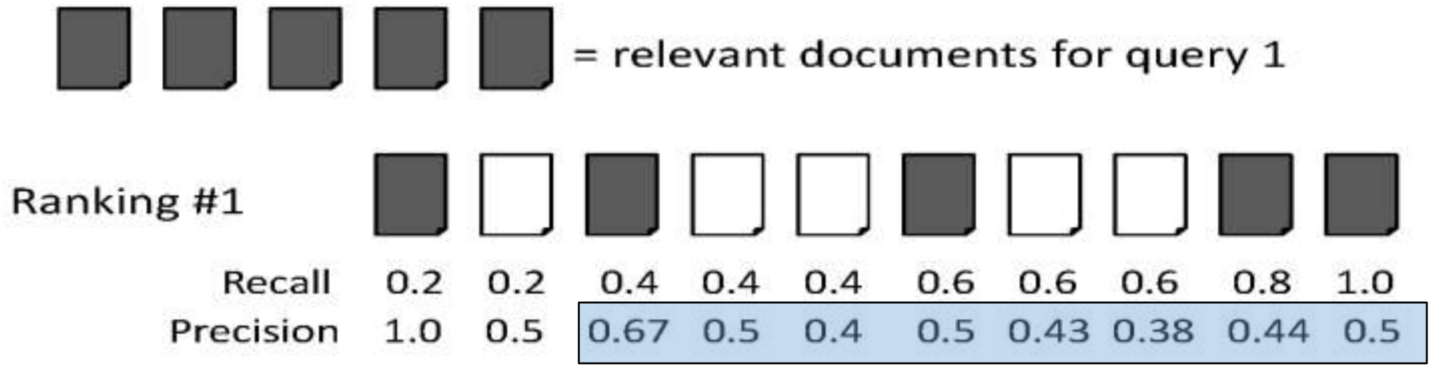
# Interpolation



| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 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 |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Recall | 0.0 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 | 1.0 |
| Interpolated Precision | 1.0 | 1.0 | 1.0 | | | | | | | | |

# Interpolation



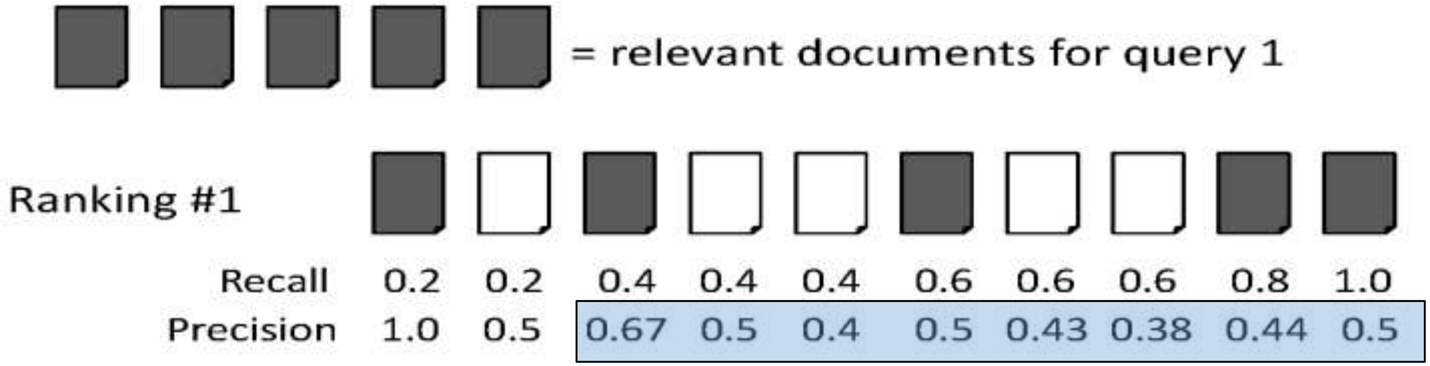
= relevant documents for query 1

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 |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Recall | 0.0 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 | 1.0 |
| Interpolated Precision | 1.0 | 1.0 | 1.0 | 0.67 | | | | | | | |

# Interpolation



| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 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 |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Recall | 0.0 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 | 1.0 |
| Interpolated Precision | 1.0 | 1.0 | 1.0 | 0.67 | 0.67 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 |

# Recap: Confusion matrix

- The confusion matrix (easily generalize to multi-class)

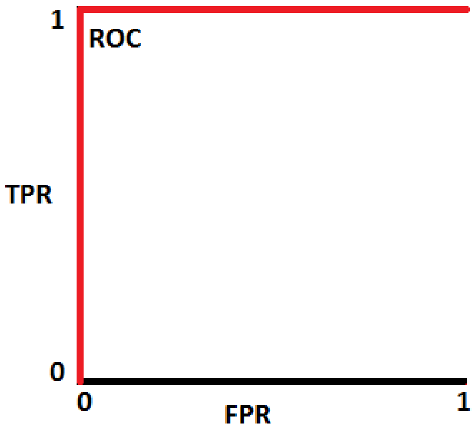
| | Actual: Positive | Actual: Negative |
|---|---|---|
| Predicted: Positive | tp | fp |
| Predicted: Negative | fn | tn |

- Machine Learning methods usually minimize FP+FN
- TPR (True Positive Rate): TP / (TP + FN) = Recall
- FPR (False Positive Rate): FP / (TN + FP)

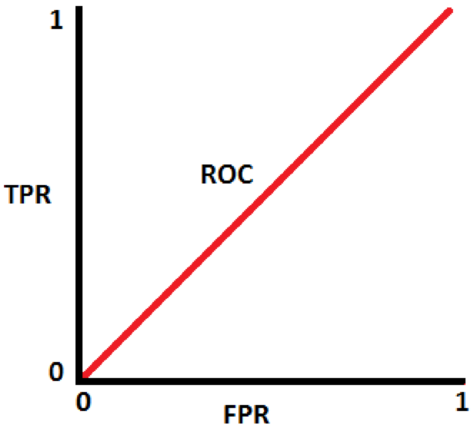
# ROC Curves

- A **receiver operating characteristic curve**, i.e. **ROC curve**, is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied.
- The diagnostic performance of a test, or the accuracy of a test to discriminate diseased cases from normal cases is evaluated using Receiver Operating Characteristic (ROC) curve analysis
- A ROC Curve is a way to compare diagnostic tests. It is a plot of the true positive rate against the false positive rate.

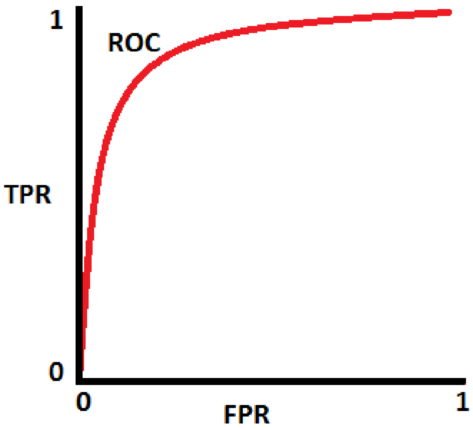
# ROC Curves



This is an ideal situation. Model has an ideal measure of separability. It is perfectly able to distinguish between positive class and negative class.

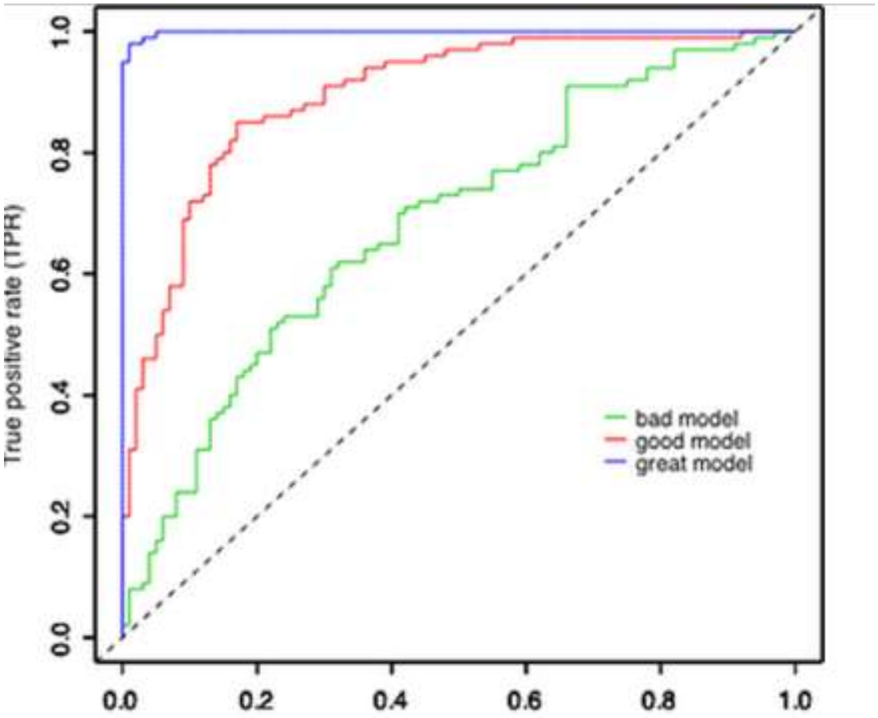


This is the worst situation. When AUC is approximately 0.5, model has no discrimination capacity to distinguish between positive class and negative class. Random predictions.



# Multiple ROC Curves

- Comparison of multiple classifiers is usually straight-forward especially when no curves cross each other. Curves close to the perfect ROC curve have a better performance level than the ones closes to the baseline.



# PR Curves Vs ROC Curves

- Remember, a ROC curve represents a relation between sensitivity (Recall) and False Positive Rate (Not Precision).
  - ROC curve plot True Positive Rate Vs. False Positive Rate; Whereas, PR curve plot Precision Vs. Recall.
- If your question is, "How well can this classifier be expected to perform *in general*, go with a ROC curve
- If true negative is not much valuable to the problem, or negative examples are abundant. Then, PR-curve is typically more appropriate.
  - For example, if the class is highly imbalanced and positive samples are very rare, then use PR-curve.
  - How meaningful is a positive result from my classifier