Homework

- Homework 1: Last chance to ask questions in person!
- Homework 2: Prepare pilot results for your final project.
  - Means you need a final project topic.
  - If you don’t have one, come talk to us.
  - If you do have one, be sure to one of us approves.

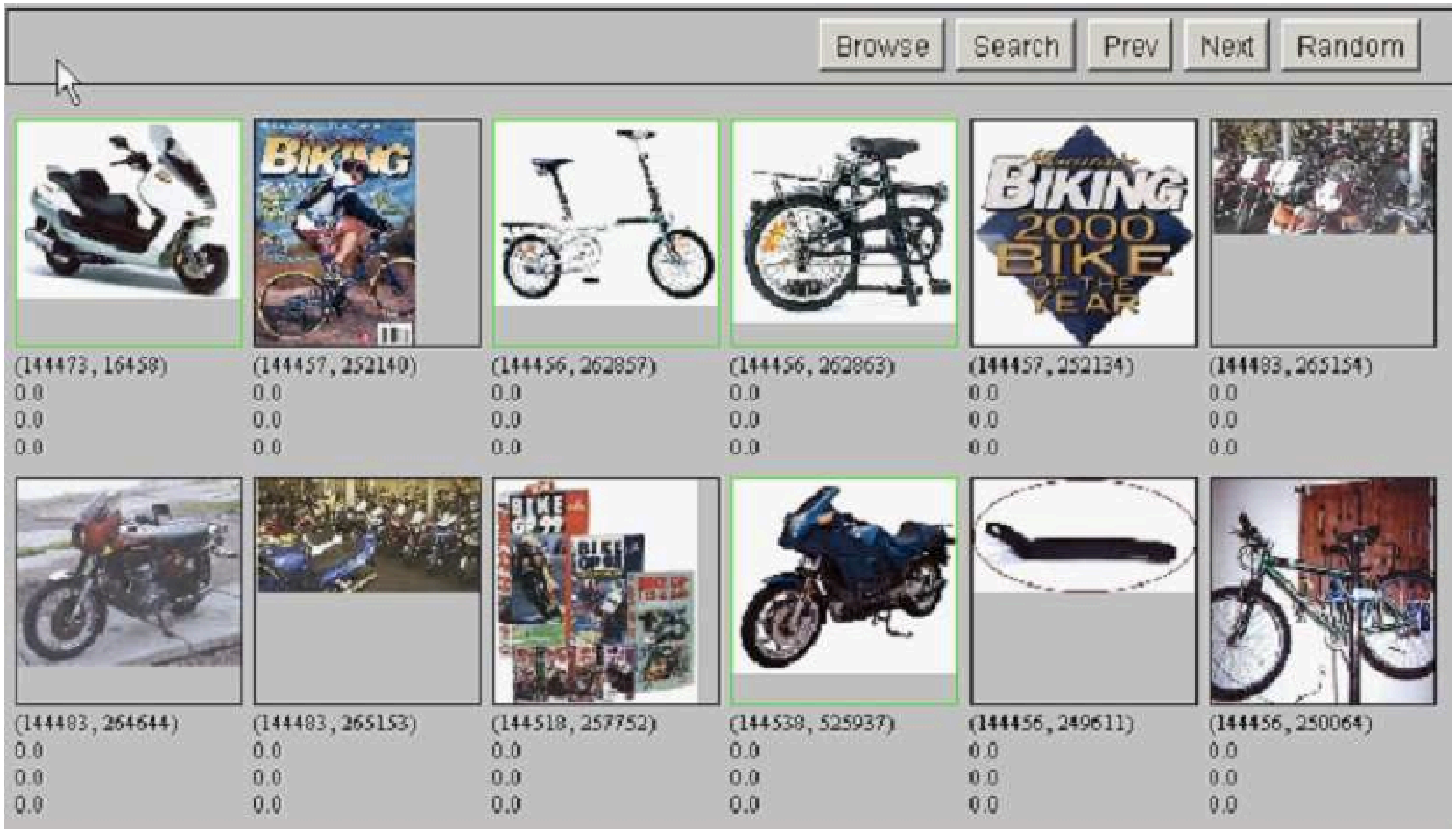
Relevance feedback (RF)

- Basic procedure:
  - User issues query.
  - System returns initial results.
  - User marks documents as relevant or irrelevant.
  - System improves representation of information need based on user's feedback.
  - System returns revised set of results.
- One of more iterations of this process.

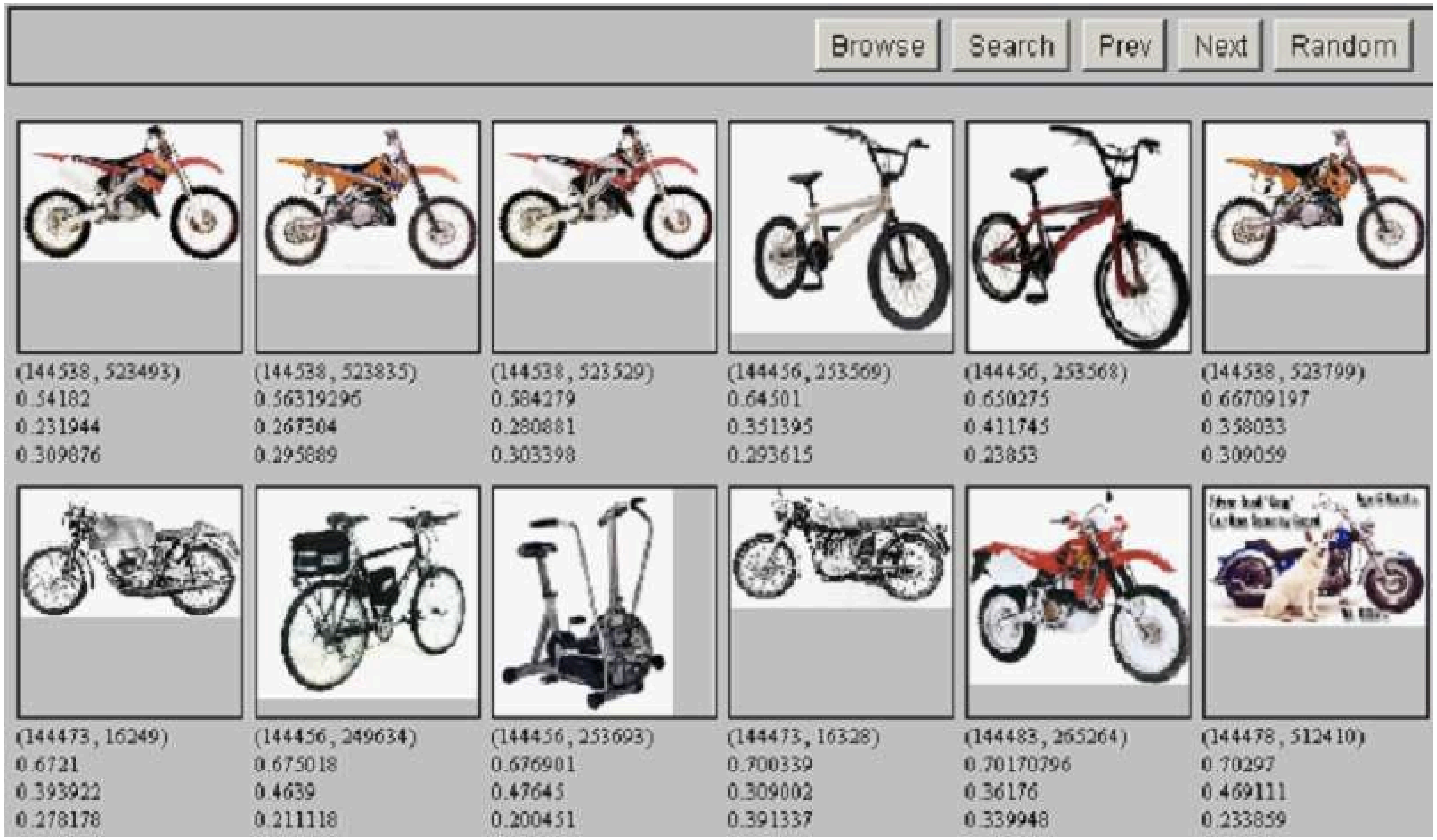
Motivation for RF

- Hard to formulate a query that retrieves exactly the documents you want.
  - Especially true if you don't know the collection well or are an inexperienced user of the system.
- Easy to look at documents and decide whether or not they match your query.
  - Even if you can't articulate what you're looking for, you'll know it when you see it.
- Also: Can help address some of the issues associated with different models of information seeking.

Example RF: Before



Example RF: After



Rocchio algorithm: Underlying theory

- Incorporates relevance feedback into vector space model.
- Find the *optimal query vector*, i.e., the one that maximizes similarity to relevant documents while minimizing similarity to non-relevant documents.

$$\vec{q}_{opt} = \arg\max_{\vec{q}} [\mathrm{sim}(\vec{q}, C_r) - \mathrm{sim}(\vec{q}, C_{nr})]$$

Rocchio algorithm: Underlying theory

- Incorporates relevance feedback into vector space model.
- Find the *optimal query vector*, i.e., the one that maximizes similarity to relevant documents while minimizing similarity to non-relevant documents.

similarity

relevant documents

non-relevant documents

$$\vec{q}_{opt} = \arg\max_{\vec{q}} [\mathrm{sim}(\vec{q}, C_r) - \mathrm{sim}(\vec{q}, C_{nr})]$$

optimal query vector

original query vector

Rocchio algorithm: Underlying theory

$$\vec{q}_{opt} = \arg\max_{\vec{q}} [\text{sim}(\vec{q}, C_r) - \text{sim}(\vec{q}, C_{nr})]$$

$$\vec{q}_{opt} = \frac{1}{|C_r|} \sum_{\vec{d}_j \in C_r} \vec{d}_j - \frac{1}{|C_{nr}|} \sum_{\vec{d}_j \in C_{nr}} \vec{d}_j$$

Rocchio algorithm: Underlying theory

$$\vec{q}_{opt} = \arg\max_{\vec{q}} [\text{sim}(\vec{q}, C_r) - \text{sim}(\vec{q}, C_{nr})]$$

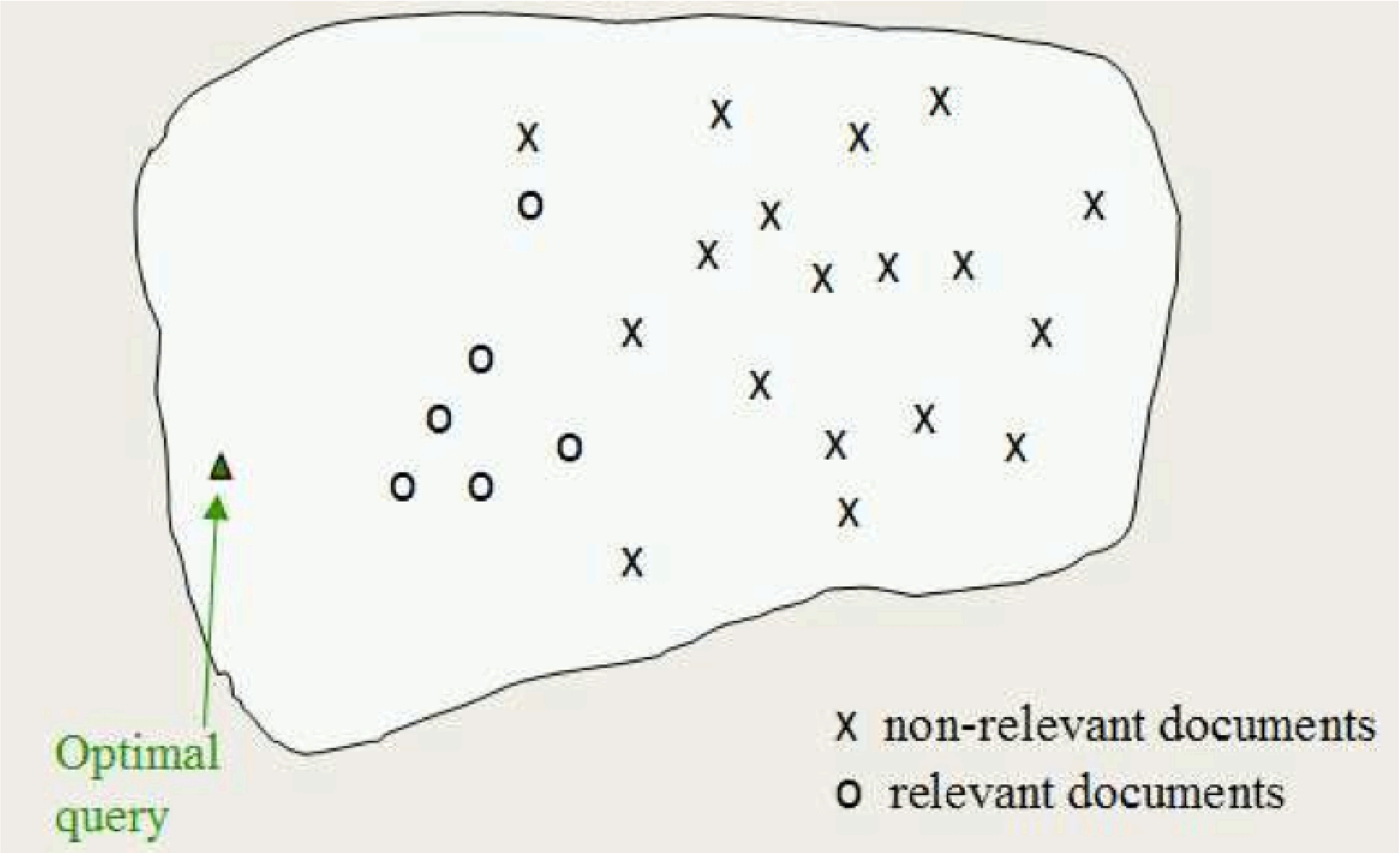
$$\vec{q}_{opt} = \frac{1}{|C_r|} \sum_{\vec{d}_j \in C_r} \vec{d}_j - \frac{1}{|C_{nr}|} \sum_{\vec{d}_j \in C_{nr}} \vec{d}_j$$

sum of all the relevant document vectors

sum of all the non-relevant document vectors

➡ Optimal query is vector difference between the centroids of the relevant and non-relevant documents.

Rocchio algorithm: Underlying theory



➡ Problem: We don't know the *full* set of relevant documents!

Rocchio algorithm

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

Rocchio algorithm

weights

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

modified query vector

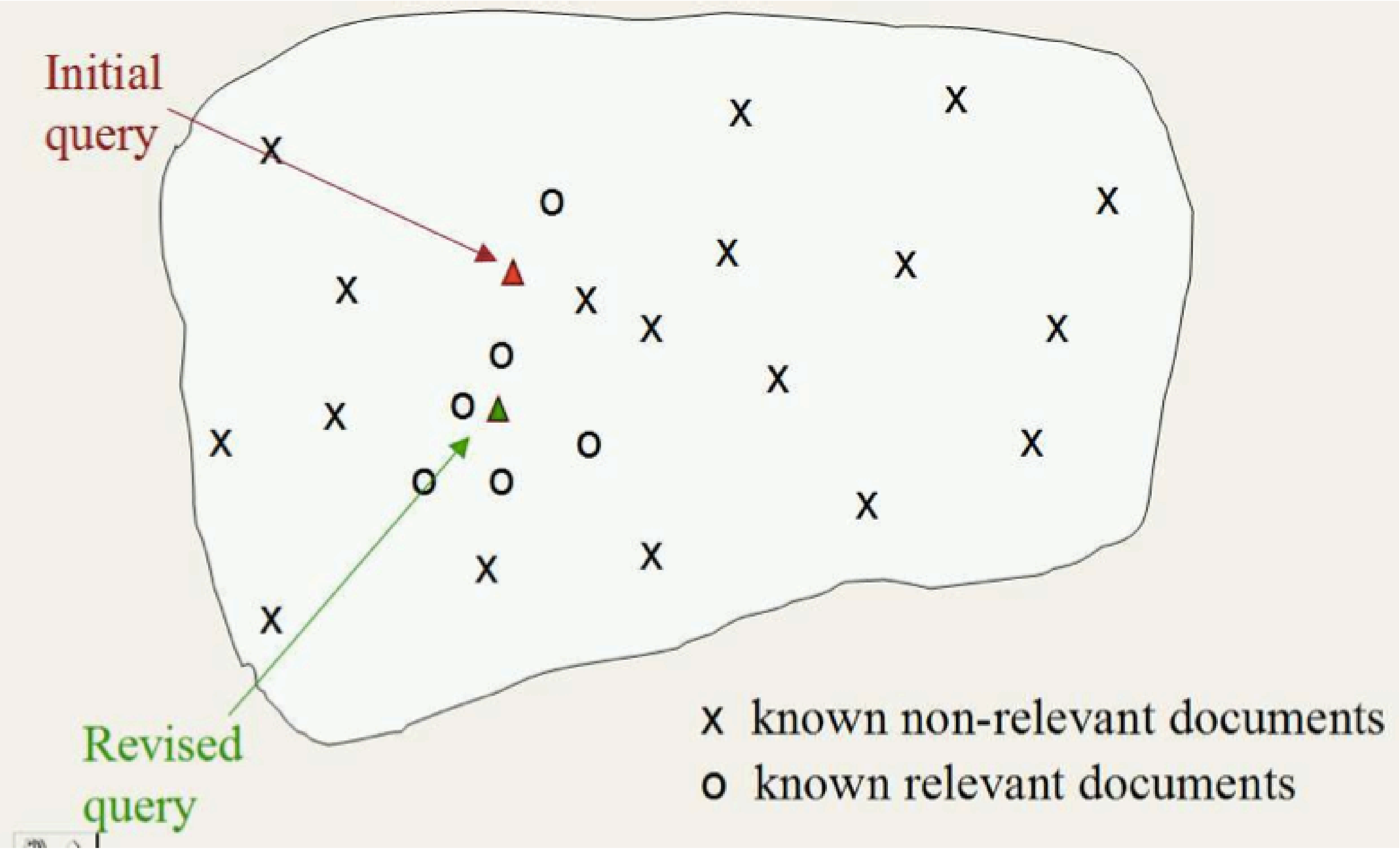
original query vector

known relevant documents

known non-relevant documents

- Weights control balance between trusting the original query and trusting the judged document set.
  - Fewer judged documents, more weight for alpha.
  - More judged documents, more weight for beta and gamma.

Rocchio algorithm: Picture example



Rocchio algorithm: Number example

new query vector = $\alpha \cdot$ original query vector +
$\beta \cdot$ relevant document vectors -
$\gamma \cdot$ non-relevant document vectors

0	4	0	8	0	0

2	4	8	0	0	2

8	0	4	4	0	16

Borrowed from: www.umiacs.umd.edu/~jimmylin/LBSC796-INFM718R.../lecture7.ppt

Rocchio algorithm: Number example

new query vector = $\alpha \cdot$ original query vector +
$\beta \cdot$ relevant document vectors -
$\gamma \cdot$ non-relevant document vectors

0	4	0	8	0	0	$\alpha = 1$
2	4	8	0	0	2	$\beta = 0.5$
8	0	4	4	0	16	$\gamma = 0.25$

Typically $\beta > \gamma$, since positive feedback is more meaningful.

Borrowed from: www.umiacs.umd.edu/~jimmylin/LBSC796-INFM718R.../lecture7.ppt

Rocchio algorithm: Number example

$$\text{new query vector} = \alpha \cdot \text{original query vector} + \beta \cdot \text{relevant document vectors} - \gamma \cdot \text{non-relevant document vectors}$$

0	4	0	8	0	0	$\alpha = 1$		0	4	0	8	0	0
2	4	8	0	0	2	$\beta = 0.5$	+	1	2	4	0	0	1
8	0	4	4	0	16	$\gamma = 0.25$	-	2	0	1	1	0	4

Typically $\beta > \gamma$, since positive feedback is more meaningful.

Borrowed from: www.umiacs.umd.edu/~jimmylin/LBSC796-INFM718R.../lecture7.ppt

Rocchio algorithm: Number example

new query vector = $\alpha \cdot$ original query vector +
$\beta \cdot$ relevant document vectors -
$\gamma \cdot$ non-relevant document vectors

0	4	0	8	0	0	$\alpha = 1$		0	4	0	8	0	0
2	4	8	0	0	2	$\beta = 0.5$	+	1	2	4	0	0	1
8	0	4	4	0	16	$\gamma = 0.25$	-	2	0	1	1	0	4
								-1	6	3	7	0	-3
								0	6	3	7	0	0

Typically $\beta > \gamma$, since positive feedback is more meaningful.

Negative term weights become 0.

Borrowed from: www.umiacs.umd.edu/~jimmylin/LBSC796-INFM718R.../lecture7.ppt

Probabilistic RF

- Alternative to reweighting the query a vector space: build a *classifier* using the user relevance feedback.
- Using a Naive Bayes classifier:

$$
\begin{aligned}
\hat{P}(x_t = 1 | R = 1) &= |VR_t| / |VR| \\
\hat{P}(x_t = 1 | R = 0) &= (df_t - |VR_t|) / (N - |VR|)
\end{aligned}
$$

Probabilistic RF

- Alternative to reweighting the query a vector space: build a *classifier* using the user relevance feedback.

- Using a Naive Bayes classifier:

probability of some term *t*, given the relevance, R, of the document

relevant documents containing *t*

relevant documents

$$
\begin{aligned}
\hat{P}(x_t = 1 | R = 1) &= |VR_t| / |VR| \\
\hat{P}(x_t = 1 | R = 0) &= (df_t - |VR_t|) / (N - |VR|)
\end{aligned}
$$

documents containing t

total number of documents

- A little more on this later, but do note that all information about the original query is lost in this formulation.

Evaluation

- Compare precision and recall of results retrieved using q_0 to those retrieved with q_m.
  - Over all documents: works great but cheating.
  - Over residual documents (excludes those assessed by user): not cheating but performance degrades, especially when there are few relevant documents.
  - Use two sets of documents: train and test.
- Other evaluation measures:
  - how does RF compare to user query reformulation?
  - how many documents found per time unit?
- Empirically, one round of RF is useful: diminishing returns on multiple iterations.

Problems with RF

- RF doesn't work well:
  - when query contains misspellings (e.g., *Mitt Rmoney*)
  - in cross-language IR scenarios
  - in document sets with vocabulary mismatch (e.g., *myocardial infarction* vs. *heart attack*)
  - when documents don't actually cluster very well
- Other issues:
  - High computing cost: queries can end up being huge.
  - Users don't like giving explicit feedback: too much work.
- Alternatives to explicit user RF: coming up next.