

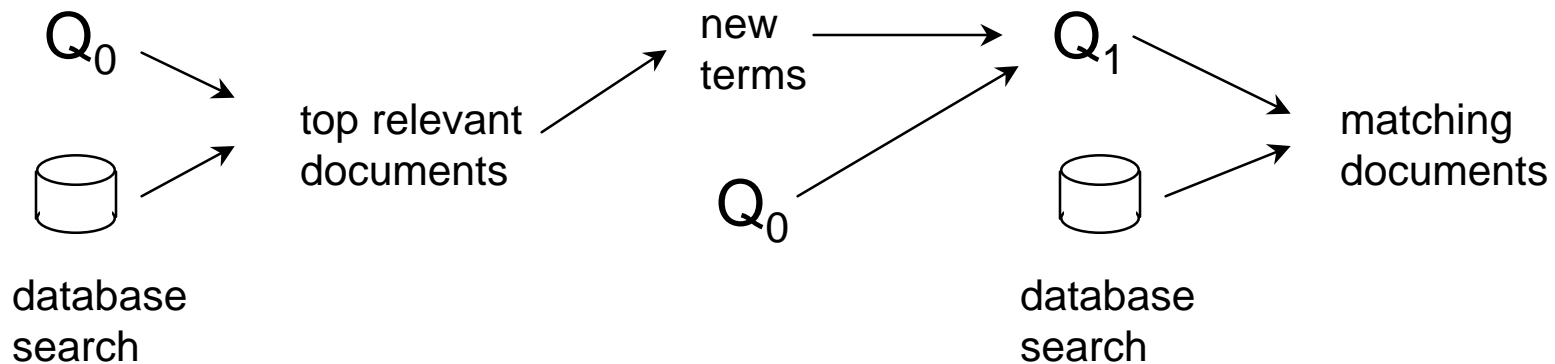
Relevance Feedback

Relevance Feedback

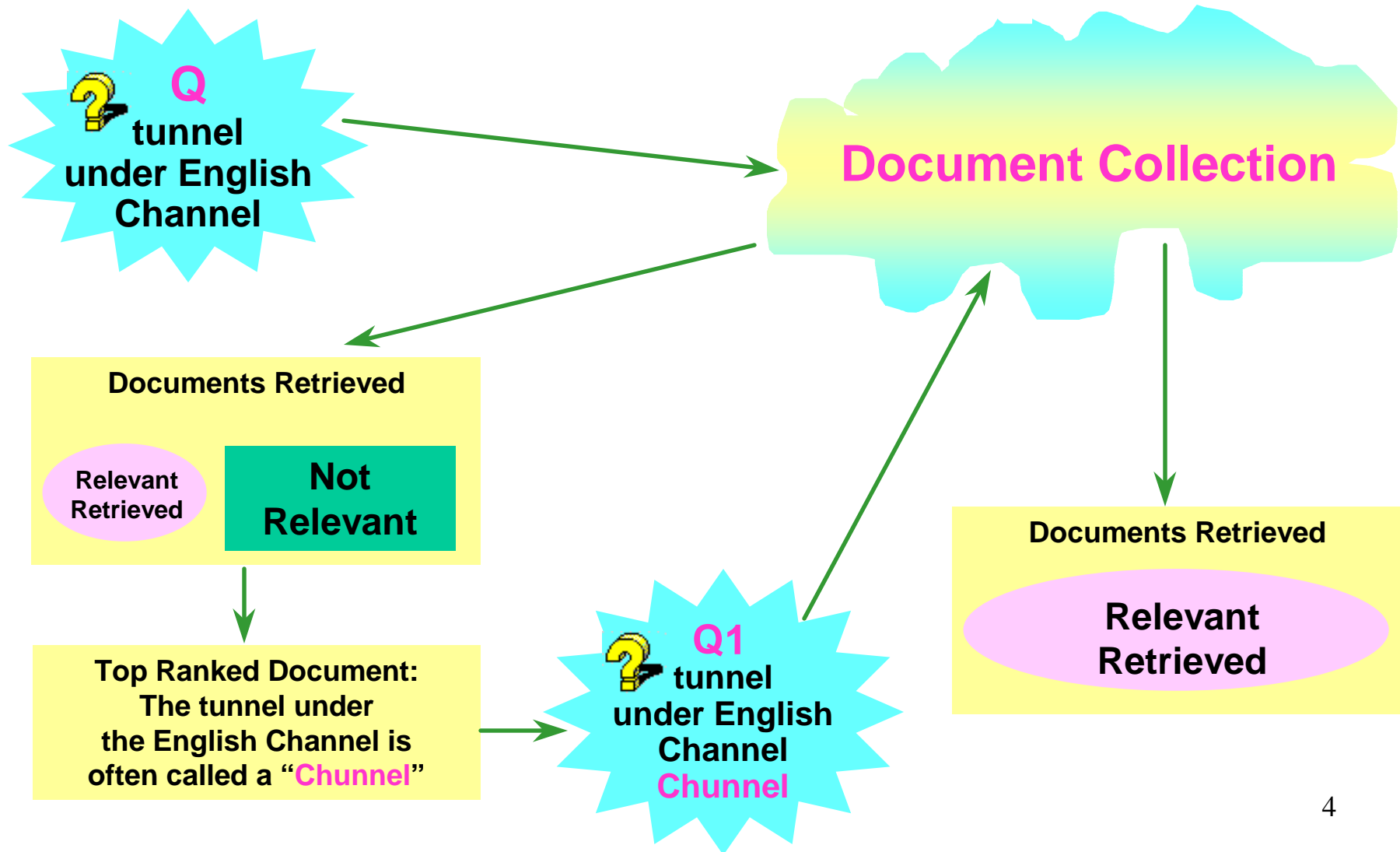
- Goal is to add terms to the query that may match relevant documents
- Basic idea is to do an initial query, get feedback from the user as to what documents are relevant and then add words from known relevant document to the query.

Relevance Feedback

- The modification of the search process to improve accuracy by incorporating information obtained from prior relevance judgments.



Relevance Feedback Example



Feedback Mechanisms

- Manual - relevant documents are identified manually and new terms are selected either manually or automatically.
- Automatic - relevant documents are identified automatically by assuming the top-ranked documents, are relevant and new terms are selected automatically.

Relevance Feedback Algorithm

- Identify the N top-ranked documents.
- Identify all terms from the N top-ranked documents.
- Select the feedback terms.
- Merge the feedback terms with the original query.
- Identify the top-ranked documents for the modified queries through relevance ranking.

Rocchio Vector Space Relevance Feedback

- $Q' = aQ + b \text{ sum}(R) - c \text{ sum}(S)$
 - Q: original query vector
 - R: set of relevant document vectors
 - S: set of non-relevant document vectors
 - a, b, c: constants (Rocchio weights)
 - Q': new query vector

Variations in Vector Model

- $Q' = aQ + b \text{ sum}(\mathbf{R}) - c \text{ sum}(\mathbf{S})$

Options:

- $a = 1, b = 1/|\mathbf{R}|, c = 1/|\mathbf{S}|$
- $a = b = c = 1$
- Use only first n documents from \mathbf{R} and \mathbf{S}
- Use only first document of \mathbf{S}
- Do not use \mathbf{S} ($c = 0$)

Overview

- Automatic (pseudo)
 - We make the leap that top ranked documents are relevant and we then add terms from these to the users query.
- Manual
 - We ask the user to tell us which documents are relevant and then we add terms from only the relevant documents to the query.

Things you can vary

- number of documents
- number of terms
- number of iterations
- weights for new terms
- phrases versus terms

Selecting Terms

- Identify top x documents
- Now for the terms in these documents, choose the top t terms.
- The *sort order* refers to how the t terms are sorted.

Sort Orders

- Harman (1988 and 1992) tried several different sort orders
- For a given term in the relevant set we have
 - n
 - number of documents that contain term t
 - f
 - number of occurrences of term t
- Sort orders
 - n
 - f
 - $n*idf$
 - $f*idf$

Example

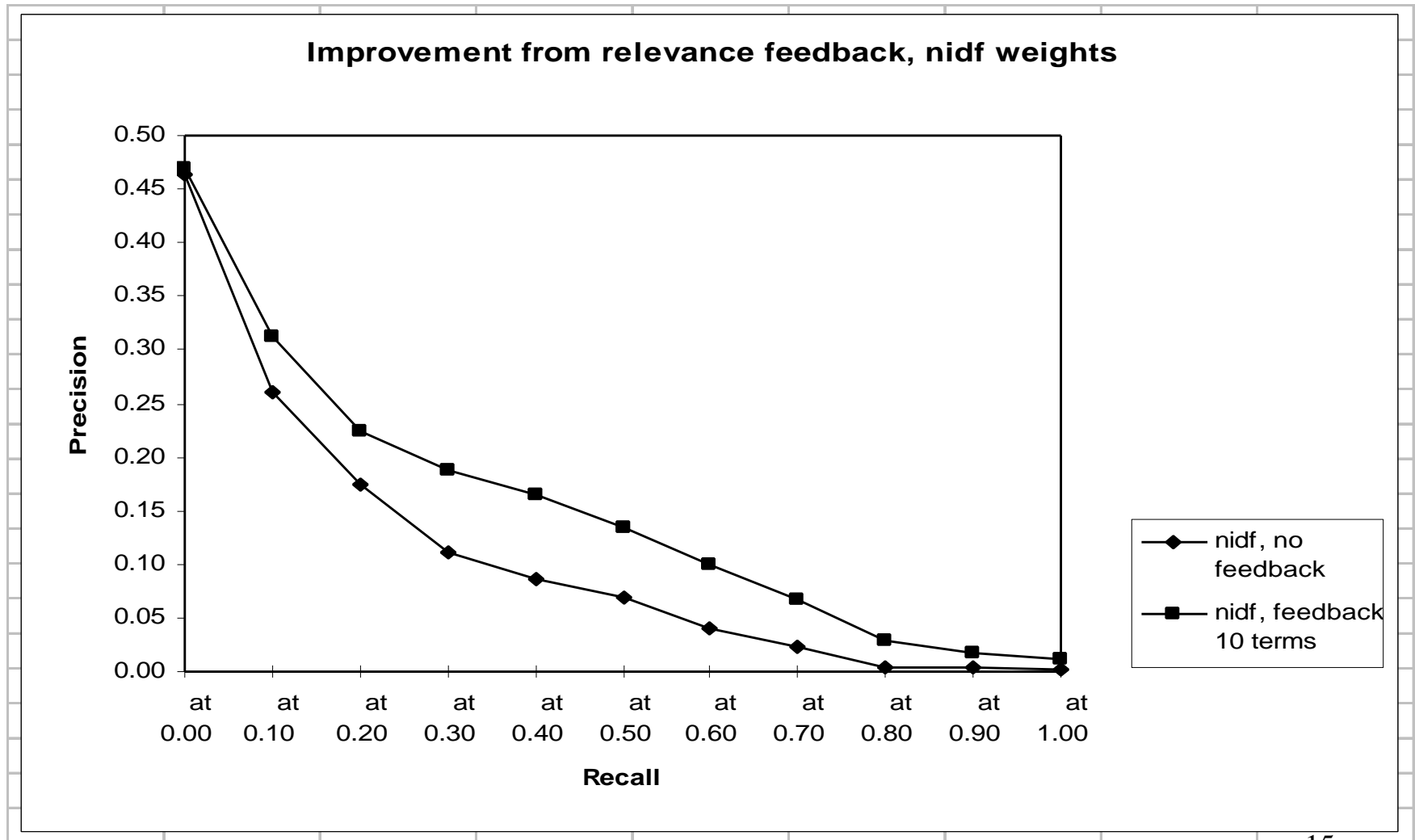
- Top 2 documents
 - d1: A, B, B, C, D
 - d2: C, D, E, E, A, A
 - d3: A, A, A
 - Assume *idf* of A, B, C is 1 and D, E is 2.

Term	n	f	n*idf	f*idf
A	3	6	3	6
B	1	2	1	2
C	2	2	2	2
D	2	2	4	4
E	1	2	4	8

Implementing Relevance Feedback

- First obtain top documents, do this with the usual inverted index
- Now we need the top terms from the top X documents.
- Two choices
 - Retrieve the top x documents and scan them in memory for the top terms.
 - Use a separate doc-term structure that contains for each document, the terms that will contain that document.

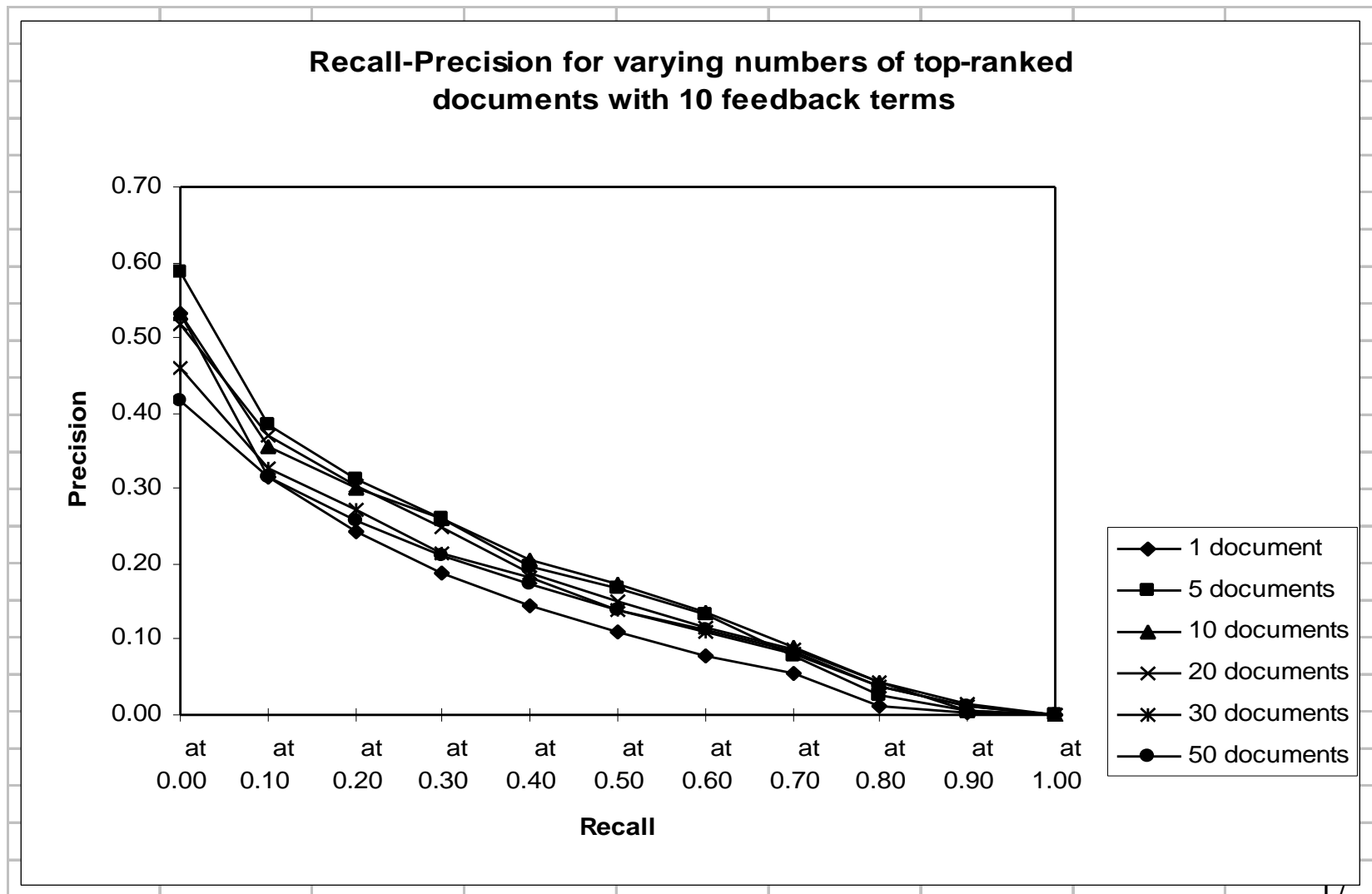
Relevance Feedback Justification



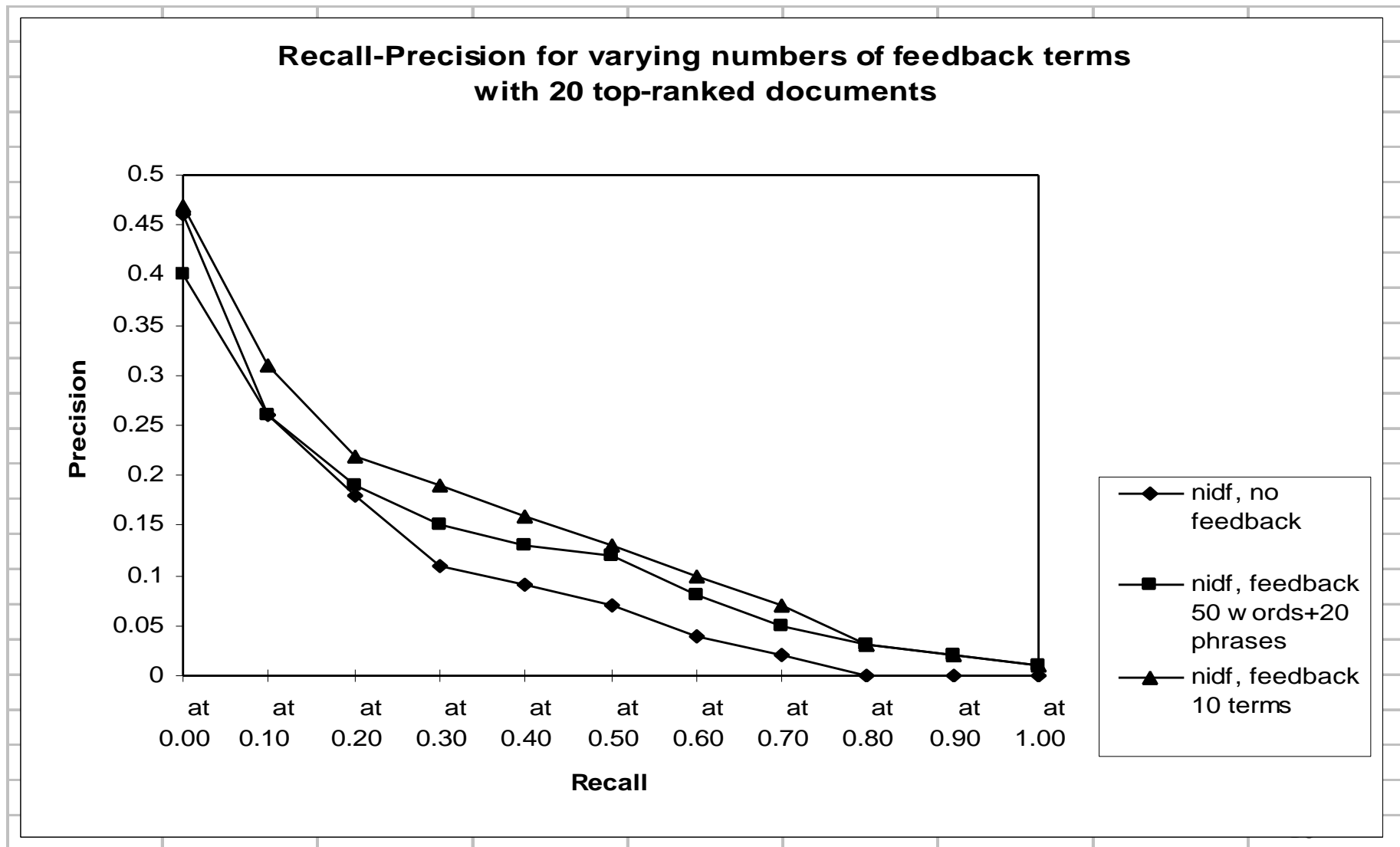
Relevance Feedback Modifications

- Various techniques can be used to improve the relevance feedback process.
 - Number of Top-Ranked Documents
 - Number of Feedback Terms
 - Feedback Term Selection Techniques
 - Term Weighting
 - Document Clustering
 - Relevance Feedback Thresholding
 - Term Frequency Cutoff Points
 - Query Expansion Using a Thesaurus

Number of Top-Ranked Documents



Number of Feedback Terms



Summary

- Pro
 - Relevance feedback usually improves average precision by increasing the number of good terms in the query (basis for the “more like this” function in Excite)
- Con
 - More computational work
 - Easy to decrease effectiveness (one horrible word can undo the good caused by lots of good words)