Learning to Find Answers to Questions on the Web

EUGENE AGICHTEIN, Columbia University

STEVE LAWRENCE, NEC Research Institute

LUIS GRAVANO, Columbia University

Presented by: Aseem Cheema

CS-856 Web Data Management

Overview:

- Introduction
- The TRITUS system
- Experimental Setup
- Evaluation Results
- Comments & Discussion

Introduction

Typical search engines treat natural language questions as lists of terms and retrieve documents similar to the original query.

What is a hard disk?

“Hard Disk: One or more rigid magnetic .. bla bla bla.. , used to store data…”

{hard disk AND “used to”}, etc.

TRITUS automatically learns to transform natural language questions into queries expected to retrieve answers to the question using a given search engine.

At run-time, TRITUS starts with a Natural Language Question and returns the documents that (are likely to) contain answers to the question.

The TRITUS System

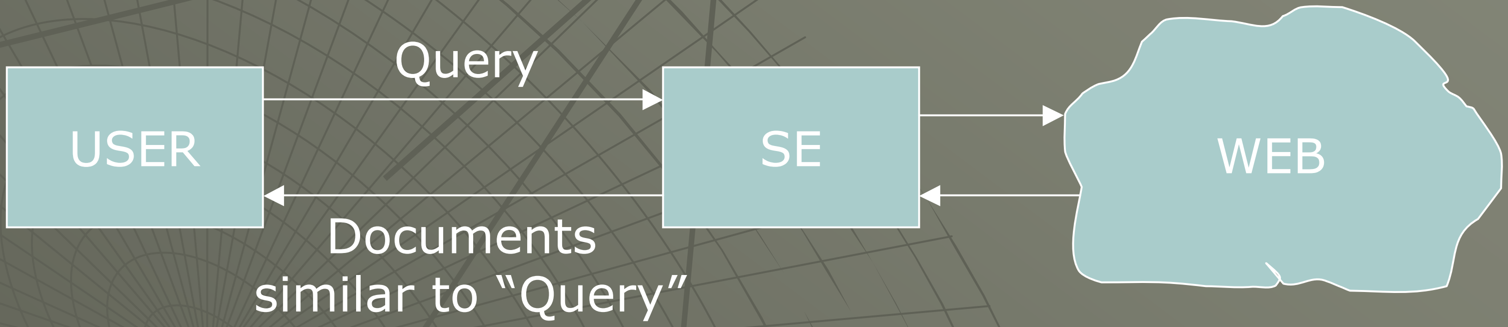
PROBLEM STATEMENT:

Retrieving a reasonable-sized set of documents that must contain an answer to a given question.

TERMS:

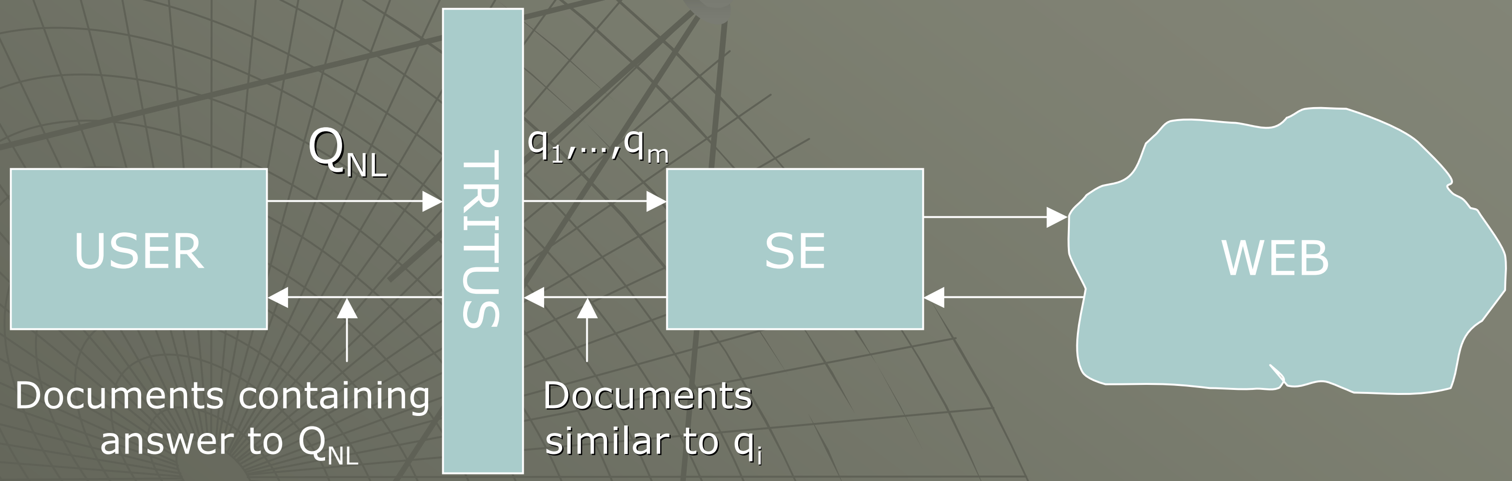
Q_{NL}	-	Natural Language Query/Question
$q_1,\ldots,q_m$	-	Queries for Q_{NL} Queries
SE	-	Search Engine (Google & Alta-Vista)
QP	-	Question Phrase
CT	-	Candidate Transform

The TRITUS System



Overview of the Problem

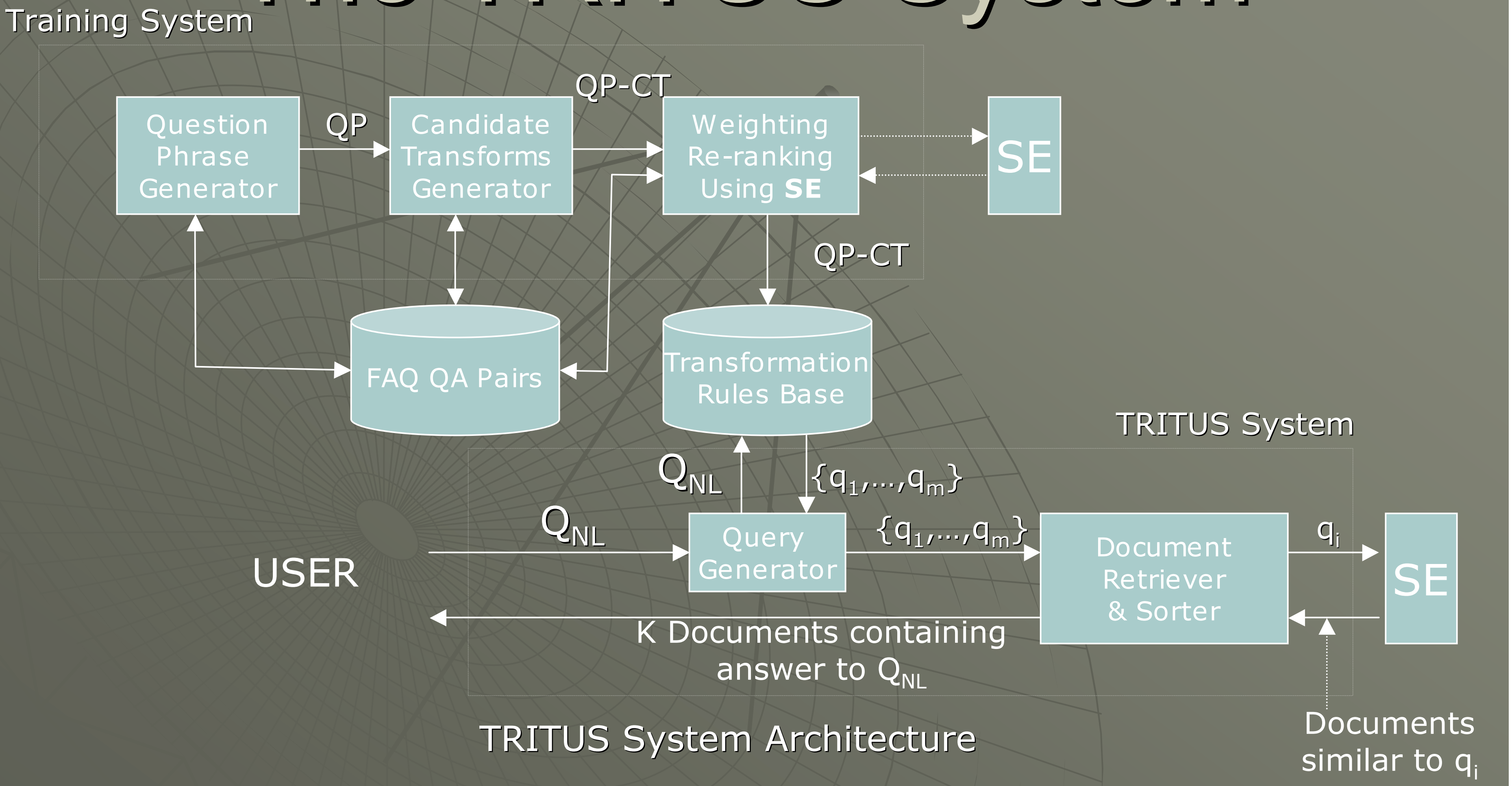
The TRITUS System



Q_{NL}
USER
TRITUS
$q_1,\dots,q_m$
SE
WEB
Documents containing answer to Q_{NL}
Documents similar to q_i

Recommended Solution

The TRITUS System


Training System
Question Phrase Generator
QP
Candidate Transforms Generator
QP-CT
Weighting Re-ranking Using **SE**
SE
QP-CT
FAQ QA Pairs
Transformation Rules Base
TRITUS System
Q_{NL}
$\{q_1,\dots,q_m\}$
Q_{NL}
USER
Query Generator
$\{q_1,\dots,q_m\}$
Document Retriever & Sorter
q_i
SE
K Documents containing answer to Q_{NL}
Documents similar to q_i

TRITUS System Architecture

Training TRAITUS

SELECTING QUESTION PHRASES

- Training Data is Question-Answer Pairs from FAQ.
- Compute frequency of all *n-grams* (phrases) of length *minQtokens* to *maxQtokens* words.
- All *n-grams* are anchored at the beginning of the question.
- At least *minQPhrCount* times.
- (^what (is|are)\s) | (^who (is|are)\s)
|(^how (do|can)\s) | (^where (is|can)\s)

Training TRAITUS

GENERATING AND FILTERING CANDIDATE TRANSFORMS

- Generate all Candidate Transforms
- Filter Candidate transforms as follows:
  - ◆ Discard CTs with nouns
  - ◆ Discard CTs with category count < catSupport
  - ◆ Keep maxPhrCount with minAPhrCount frequency
- Apply IR techniques to calculate term weights to rank the Candidate Transforms

Training TRAITUS

Generate all Candidate Transforms

- Each pair is also assigned a FAQCategory
- Tagged collection with part of speech [Brill 1992]
- For each pair of QA, where prefix of Q matches QP, all possible potential answers are generated.
- All *n-grams* of length *minAtokens* to *maxAtokens* words, starting at every word boundary in the first *maxLen* bytes of the Answer text.

The TRITUS System

Training TRAITUS

Filtering Candidate Transforms

- Discard CTs with nouns – This is done to avoid changing the intended topic of the query.
- Discard CTs with category count < catSupport – This is done to avoid domain specific transforms.
- Keep maxPhrCount with minAPhrCount frequency – Done to make the following steps computationally less expensive.

Training TRAITUS

Weighting and Ranking of Candidate Transforms

Relevance Based Term Weight *(w_i)* for tr_i

$$w_i = \log \frac{(r + 0.5) / (R - r + 0.5)}{(n - r + 0.5) / (N - n - R + r + 0.5)}$$

Co-occurrence count of tr_i with QP *(qtf_i)*

Term selection weight of tr_i *(wtr_i)*

$wtr_i = qtf_i \,.\, w_i$

The TRITUS System

Training TRAITUS

Weighting and Ranking of Candidate Transforms

Relevance Based Term Weight *(w_i)* for tr_i

$$w_i = \log \frac{(r + 0.5) / (R - r + 0.5)}{(n - r + 0.5) / (N - n - R + r + 0.5)}$$

	Relevant	Non-Relevant	
Containing the term	r	n-r	n
Not containing the term	R-r	N-n-R+r	N-n
	R	N-R	N

N – number of documents in the collection.
n – number of documents containing term.
R – number of relevant documents.
r – number of relevant documents containing term.

Training TRAITUS

Weighting and Ranking of Candidate Transforms

Candidate Transforms are sorted into buckets

Transform Length	CT tr_i	wtr_i
3	“is used to” “according to the” “to use a”	32.89 23.49 21.43
2	“is a” “of a” “refers to”	298.89 94.34 81.3
1	“usually” “used” “refers”	128.23 110.39 80.1

Question Phrase “what is a”

The TRITUS System

Training TRAITUS

Weighting and Re-Ranking using Search Engines

Algorithm for ranking a set of CTs for single QP and SE.
Procedure is repeated for all SEs and QPs.
Evaluate performance of each CT on Web Search Engines

Step1: Retrieve a set of QA pairs uniformly from FAQ Categories.

Step2: {QP C} to {C [AND,NEAR,..] tr_i }
Stop Word dictionary generated.

Step3: Top 10 documents retrieved using Search Engine.
These documents are analyzed in the following steps.

Training TRAITUS

Weighting and Re-Ranking using Search Engines

Step4(a): Document is broken into subdocuments.
if *subDocLen* = N, then starting positions are 0, N/2, N, 3.N/2, …..

Step4(b): *docScore*(Answer,D)=Max_i(BM25$_{\text{phrase}}$(Answer,SD$_i$))

$$BM25_{phrase} = \sum_{i=0}^{|Q|} w_i \frac{(k_1+1)tf_i\ (k_3+1)qtf_i}{(K+tf_i)(k_3+qtf_i)}$$

$k_1 = 1.2$, $k_3 = 1000$, $k = k_1((1-b)+b.dl/avdl)$, $b=0.5$

dl is document length, *avdl* is average document length in terms

tf_i is term frequency in the document.

The TRITUS System

Training TRAITUS

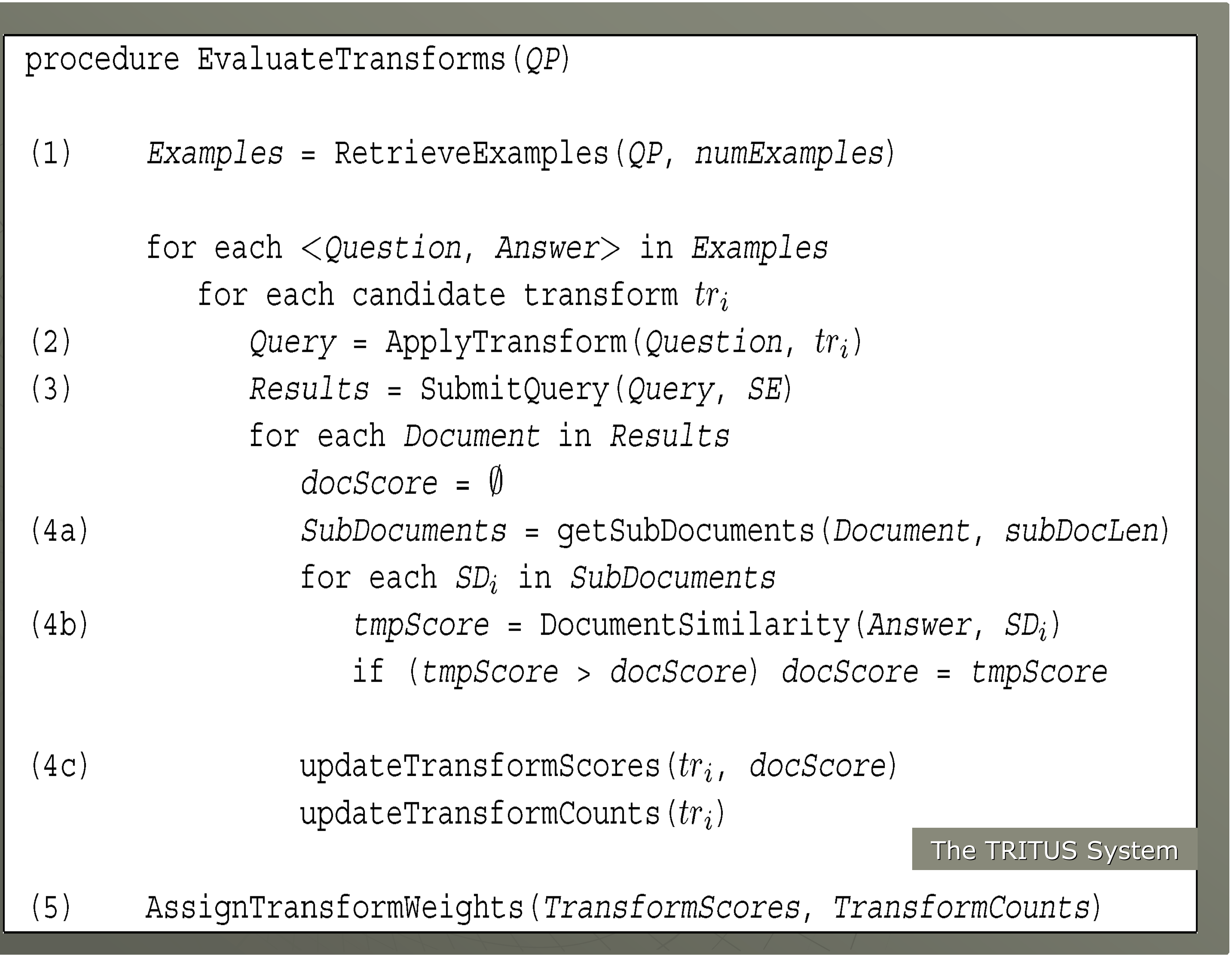
Weighting and Re-Ranking using Search Engines

The weight for a term or a phrase t is calculated as follows:

$$w = \left\{ \begin{array}{ll} w_t & \text{if } w_t \text{ is defined for } t \\ \log IDF(t) & \text{if } w_t \text{ is not defined but } IDF(t) \text{ is} \\ NumTerms(t). \sum_{t_i \in t} \log IDF(t_i) & \text{otherwise} \end{array} \right\}$$

Step5:Weight WT_i of transform tr_i is the average similarity between the original training answers and the documents returned.

$$WT_i = \frac{\sum_{<Q,A>} docScore(A, D_{tr_i})}{Count(D_{tr_i})}$$



```
procedure EvaluateTransforms(QP)

(1)   Examples = RetrieveExamples(QP, numExamples)

      for each <Question, Answer> in Examples
         for each candidate transform $tr_i$
(2)         Query = ApplyTransform(Question, $tr_i$)
(3)         Results = SubmitQuery(Query, SE)
            for each Document in Results
               docScore = $\emptyset$
(4a)           SubDocuments = getSubDocuments(Document, subDocLen)
               for each $SD_i$ in SubDocuments
(4b)              tmpScore = DocumentSimilarity(Answer, $SD_i$)
                  if (tmpScore > docScore) docScore = tmpScore

(4c)           updateTransformScores($tr_i$, docScore)
               updateTransformCounts($tr_i$)

(5)   AssignTransformWeights(TransformScores, TransformCounts)
```

The TRITUS System

TRAITUS in action

Run-Time query Reformulation

Reformulate question with preference for longer phrases.
Corresponding Transforms and their weights are retrieved.
Only *numTransforms* transforms are used.
All the documents using all the transforms are retrieved.

CommonTerms returns the number of non-stop terms common between the transformed query and the subdocument.

Score is calculated for each document
(max$_{CommonTerms}$*) X (Weight of tr$_i$) => Incremental if documents returned by more than one tr$_i$.*

Ranking is done based on the score and K top ranked documents are returned.

The TRITUS System

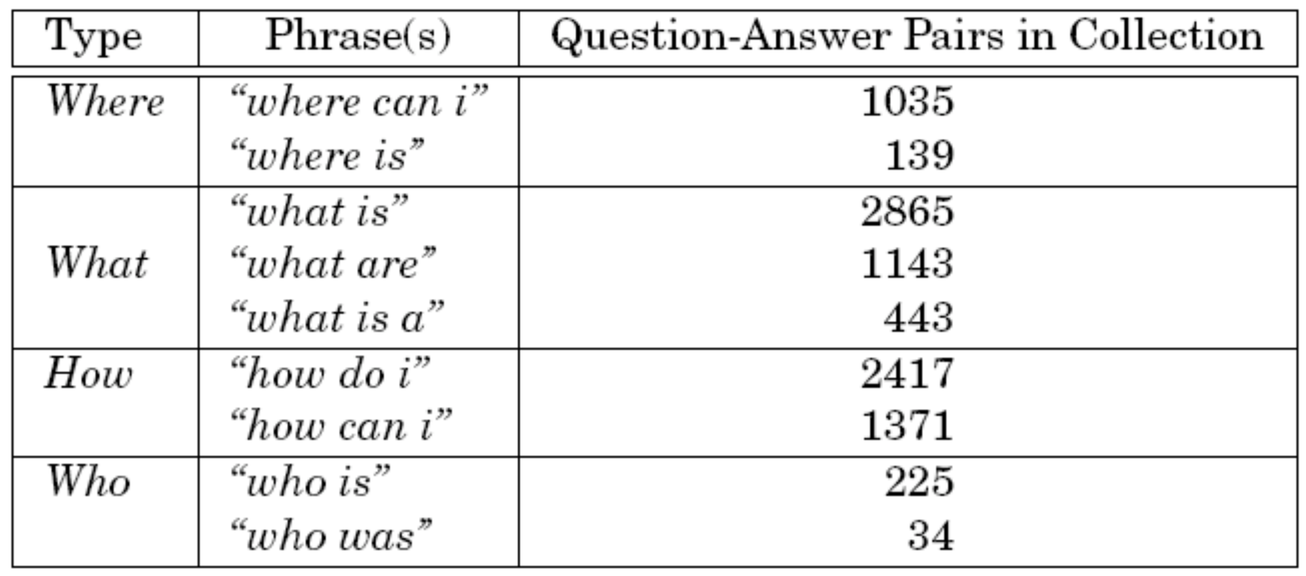
```
procedure EvaluateQuestion($Question$, $K$)

(1a) $QP$ = matchQuestionPhrase($Question$)
(1b) ($tr$,$WT$) = retrieveTransforms($QP$, $numTransforms$)
(1c) $Results$=$\emptyset$, $Documents$=$\emptyset$, $Scores$=$\emptyset$

     for each $tr_i$ in $tr$
(2)     $Query$ = ApplyTransform($Question$, $tr_i$)
(3a)    $Results_i$ = SubmitQuery($Query$, $SE$)
(3b)    $Documents$ += $Results_i$, $Results$ += $Results_i$

     for each $Results_i$ in $Results$
        for each document $d_j$ in $Results_i$
(4a)       $SubDocuments$ = getSubDocuments($d_j$, $subDocLen$)
           for each $SD_k$ in $SubDocuments$
(4b)          $tmpScore_k$ = CommonTerms($Query$, $SD_k$)
(4c)       $Scores_j$ += $\text{Max}_k(tmpScore_k) \cdot WT_i$

(5)  $RankedDocuments$ = Sort $Documents$ in decreasing order of $Scores$
(6)  Return the $K$ $RankedDocuments$ with highest $Scores$
```

The TRITUS System

EXPERIMENTAL SETUP

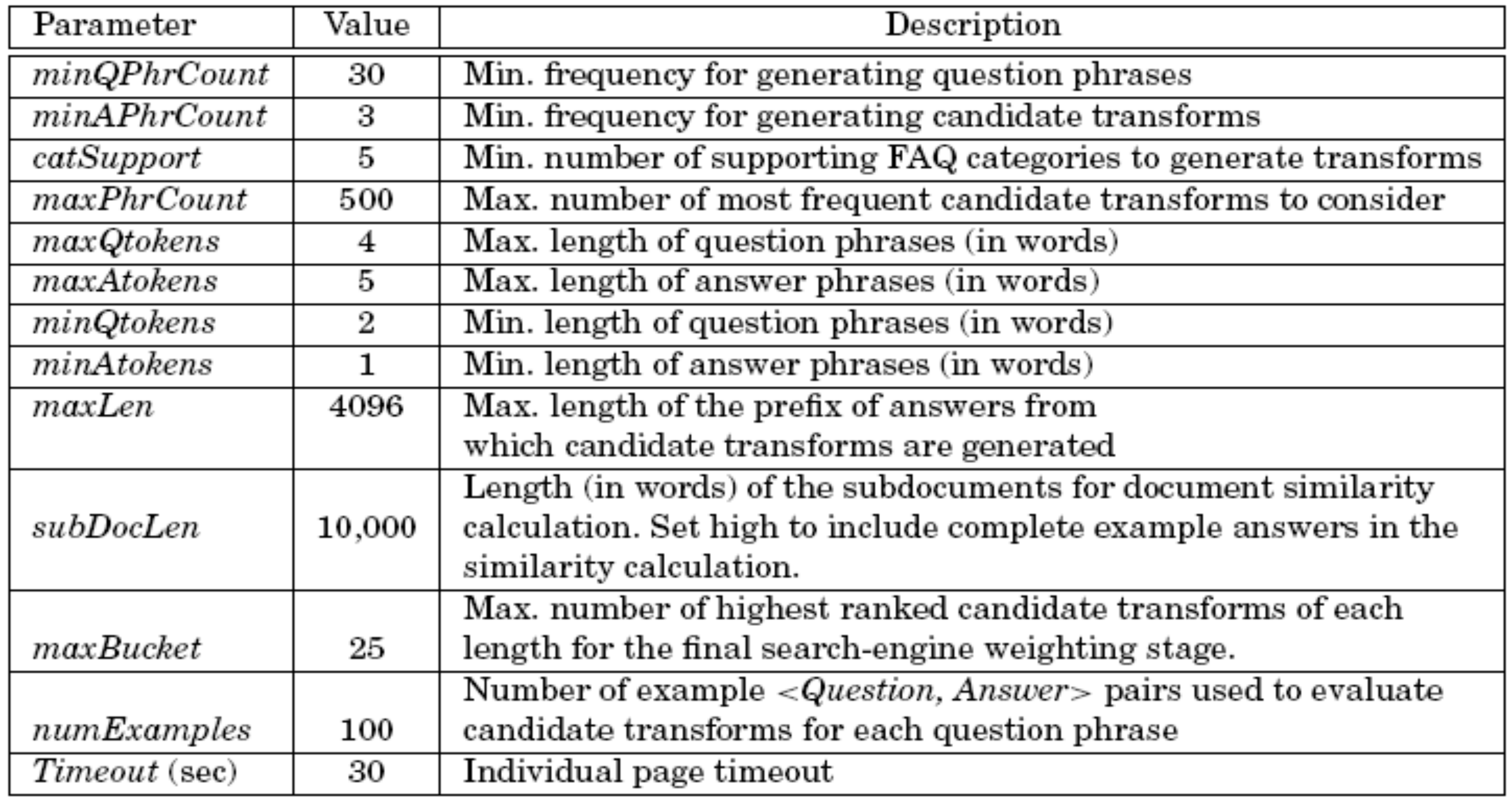
Training TRITUS

30,000 QA pairs from 270 FAQ files on various subjects.



Type	Phrase(s)	Question-Answer Pairs in Collection
Where	*"where can i"*	1035
	"where is"	139
What	*"what is"*	2865
	"what are"	1143
	"what is a"	443
How	*"how do i"*	2417
	"how can i"	1371
Who	*"who is"*	225
	"who was"	34

EXPERIMENTAL SETUP



Parameter	Value	Description
minQPhrCount	30	Min. frequency for generating question phrases
minAPhrCount	3	Min. frequency for generating candidate transforms
catSupport	5	Min. number of supporting FAQ categories to generate transforms
maxPhrCount	500	Max. number of most frequent candidate transforms to consider
maxQtokens	4	Max. length of question phrases (in words)
maxAtokens	5	Max. length of answer phrases (in words)
minQtokens	2	Min. length of question phrases (in words)
minAtokens	1	Min. length of answer phrases (in words)
maxLen	4096	Max. length of the prefix of answers from which candidate transforms are generated
subDocLen	10,000	Length (in words) of the subdocuments for document similarity calculation. Set high to include complete example answers in the similarity calculation.
maxBucket	25	Max. number of highest ranked candidate transforms of each length for the final search-engine weighting stage.
numExamples	100	Number of example <*Question, Answer*> pairs used to evaluate candidate transforms for each question phrase
Timeout (sec)	30	Individual page timeout

EXPERIMENTAL SETUP

Retrieval Systems Compared

TREC QA evaluation was not used because answers are not being retrieved and TRITUS is more general purpose system.

1. Google **(GO)** search engine.
2. TRITUS optimized for Google **(TR-GO)**
3. AltaVista **(AV)** search engine.
4. TRITUS optimized for AltaVista **(TR-AV).**
5. Tritus over both Google and AltaVista **(TR-ALL).**
6. AskJeeves **(AJ).**

EXPERIMENTAL SETUP

Evaluation Metrics

PRECISION: %age of relevant documents

HELPFULLNESS: %age of questions where system performs the best.

MEAN RECIPROCAL RANK: Reciprocal rank is reciprocal of highest rank of a relevant document. Mean Reciprocal Rank is the average of Reciprocal Ranks for all evaluated queries.

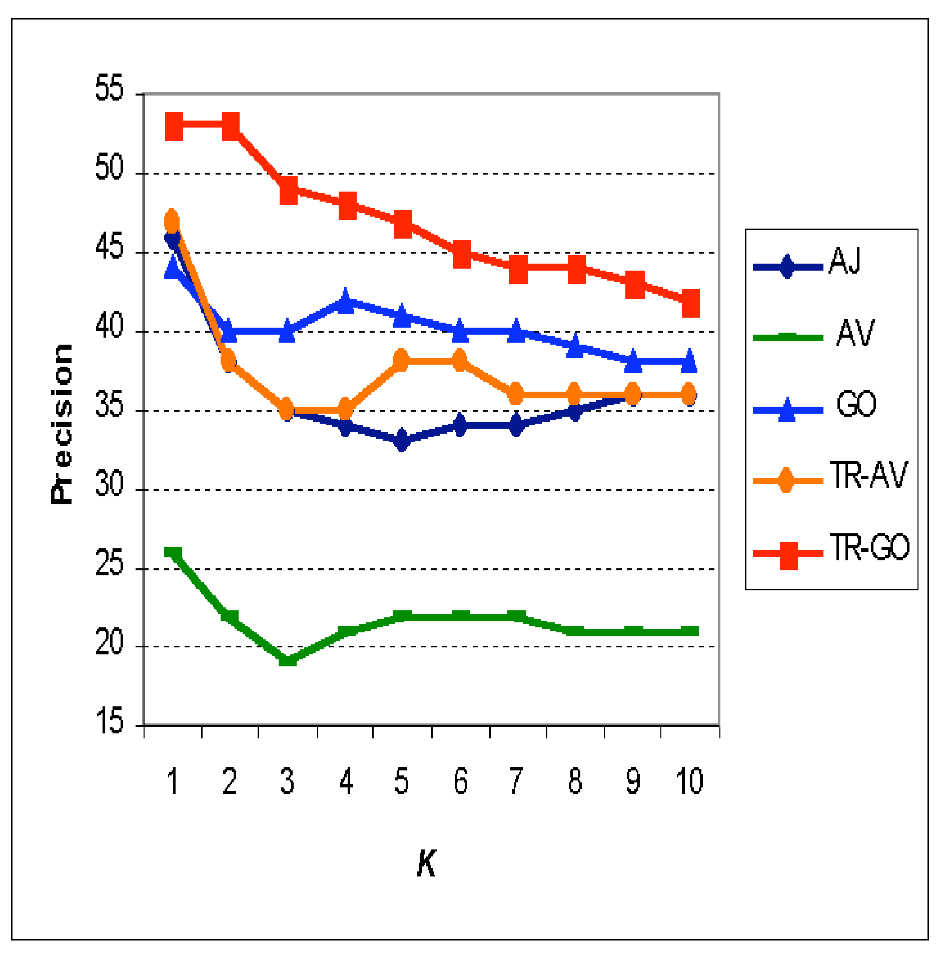
EXPERIMENTAL SETUP

Evaluation Queries and Their Relevance Judgments

1. Real User questions from the log of queries received by the Excite search engine on 20th December, 1999.
2. 2.5 million queries, 290,000 Natural Language Questions.
3. 90% questions estimated to be Where, What, How & When.
4. Random sample of 50 questions chosen for each type
5. Top 10 URLs are retrieved for each System and mixed for the volunteers to evaluate.
6. Volunteers are blind to the System.
7. Page is “good”, “bad” or “ignore”.

Evaluation Results

Year 2000 Evaluation

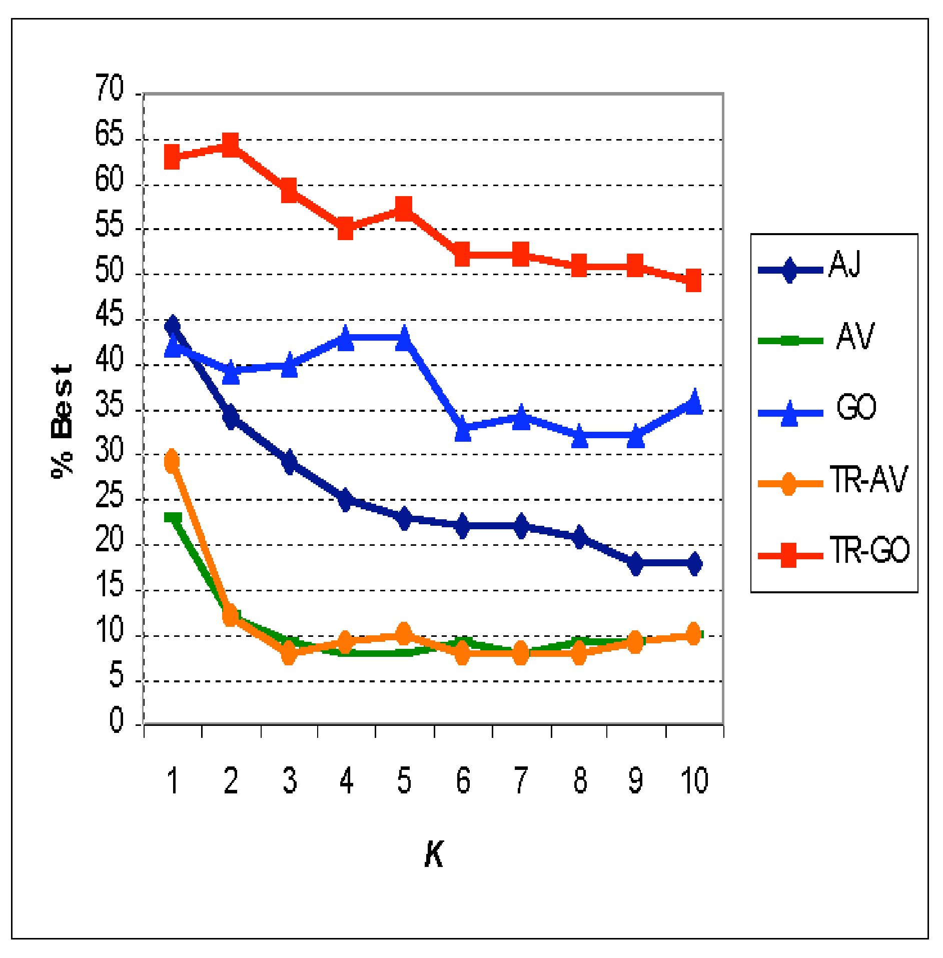


Average *precision at* K over 89 test queries, for varying number of top documents examined K.

Tritus consistently outperforms the underlying search engine that it is based on, and Tritus-Google is the best performing system

Evaluation Results

Year 2000 Evaluation



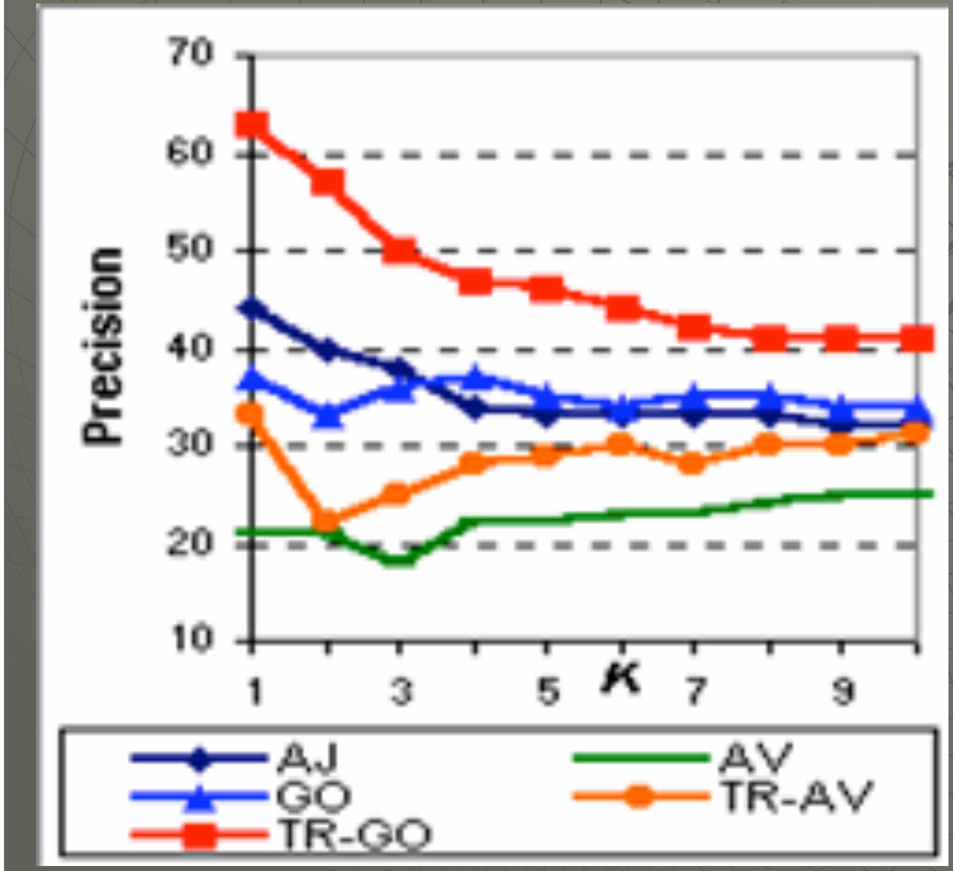
HELPFULLNESS over 89 test queries.

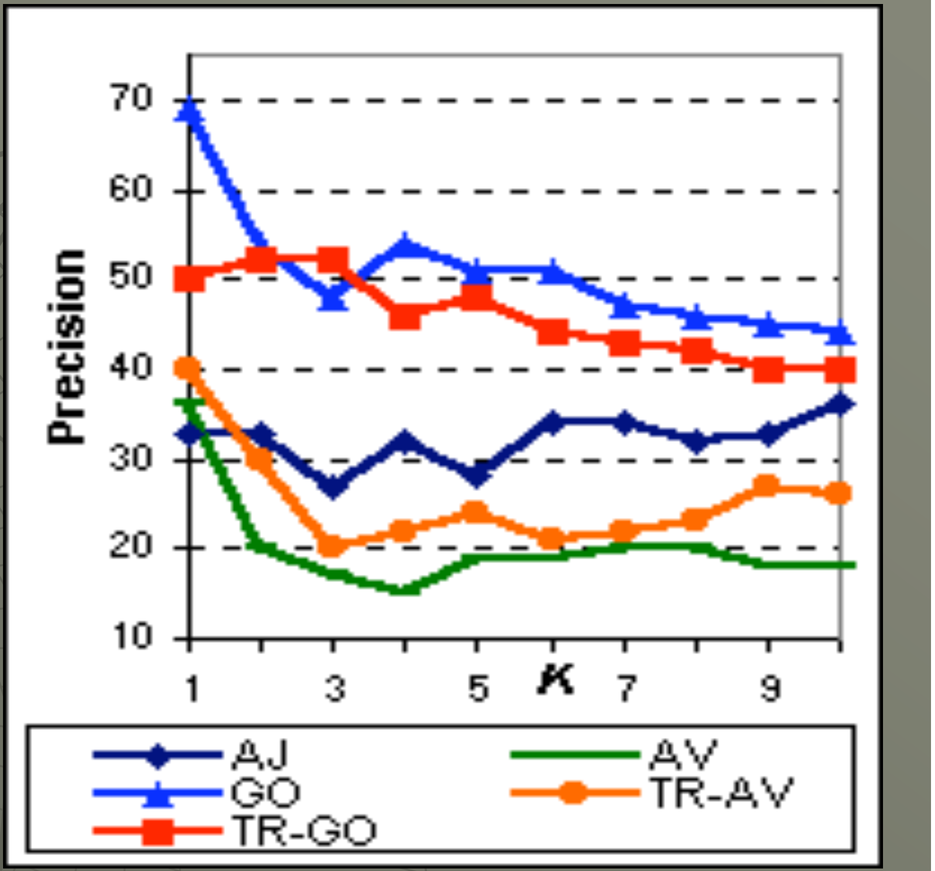
Multiple systems can return most relevant.

Lower performing systems on this metric not very meaningful.

Evaluation Results

Year 2000 Evaluation

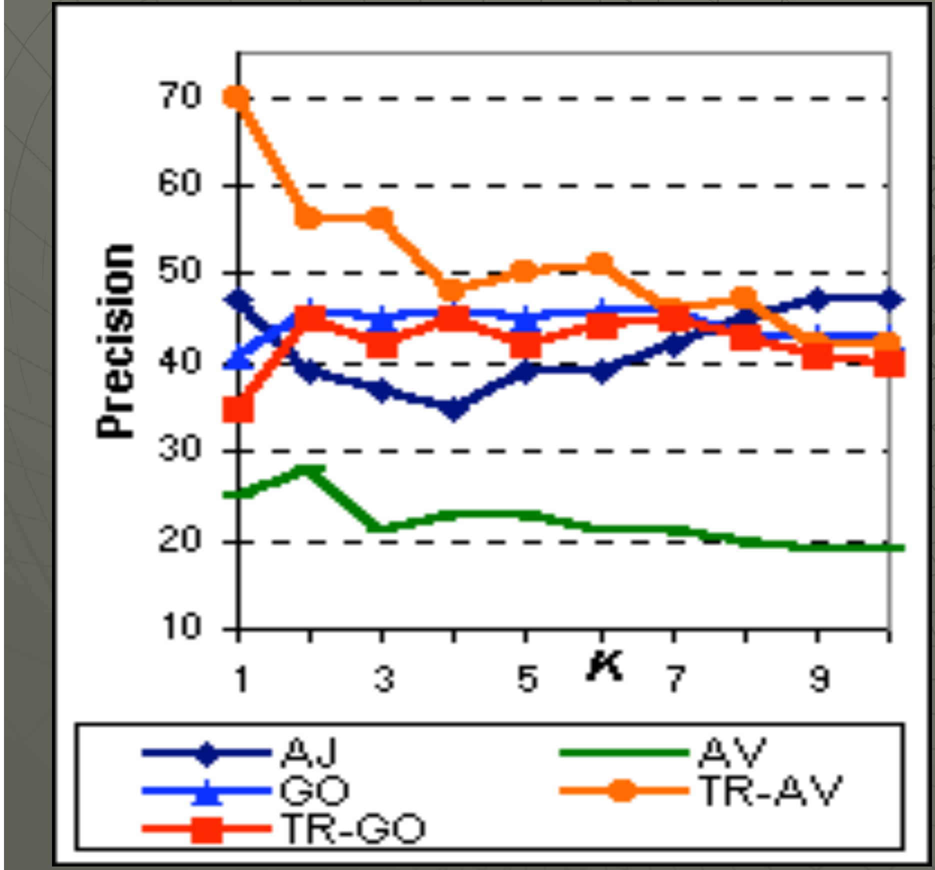


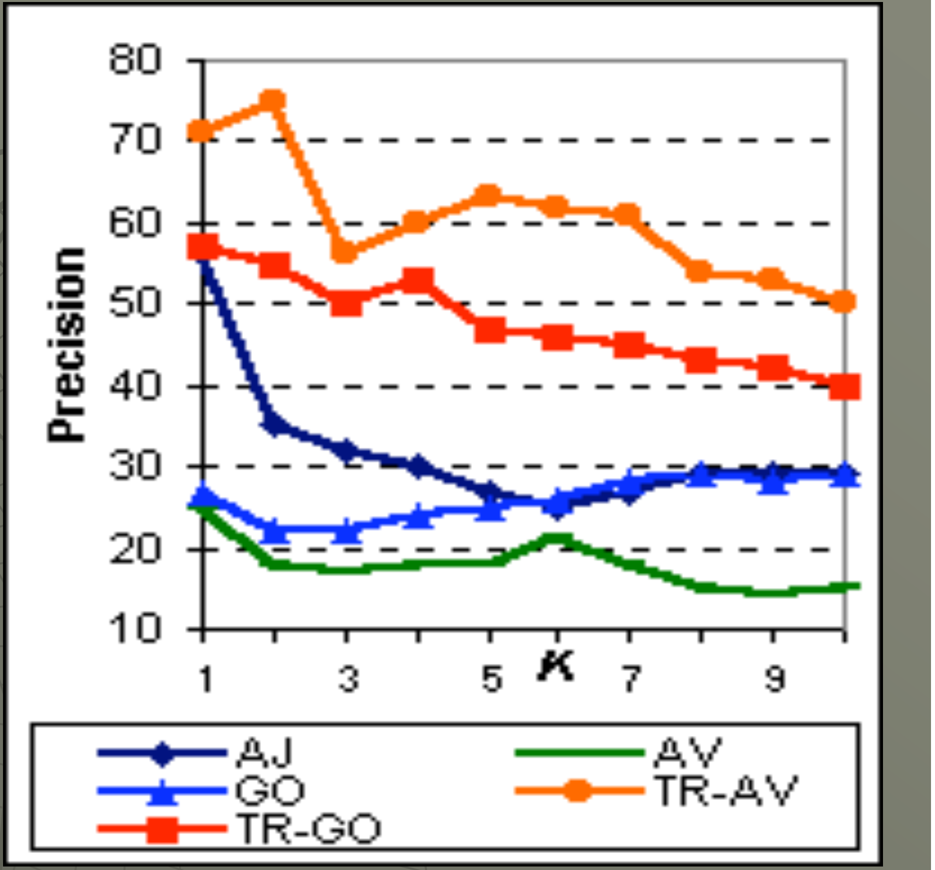


What **How**

Evaluation Results

Year 2000 Evaluation

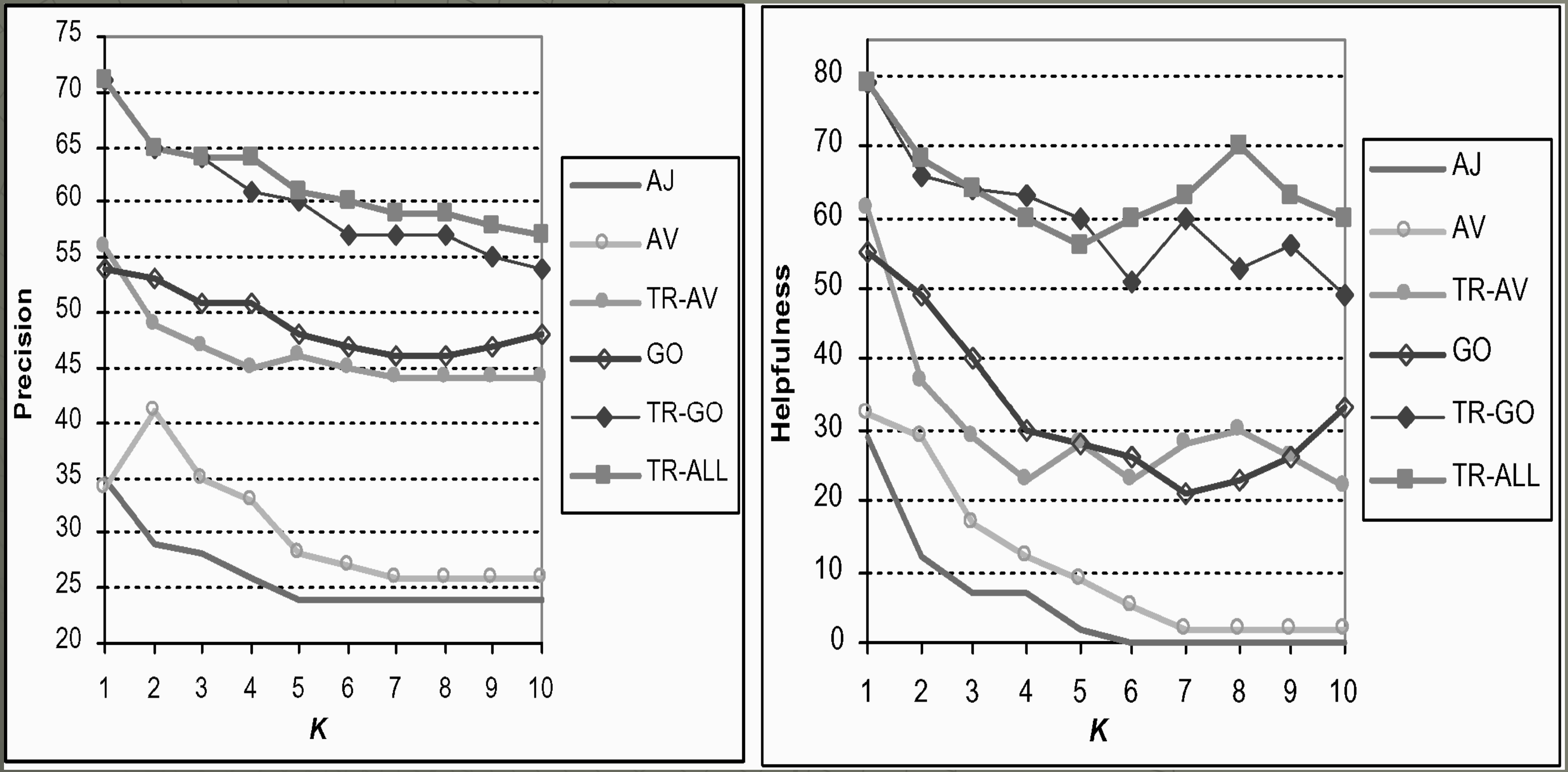




Where **Who**

Evaluation Results

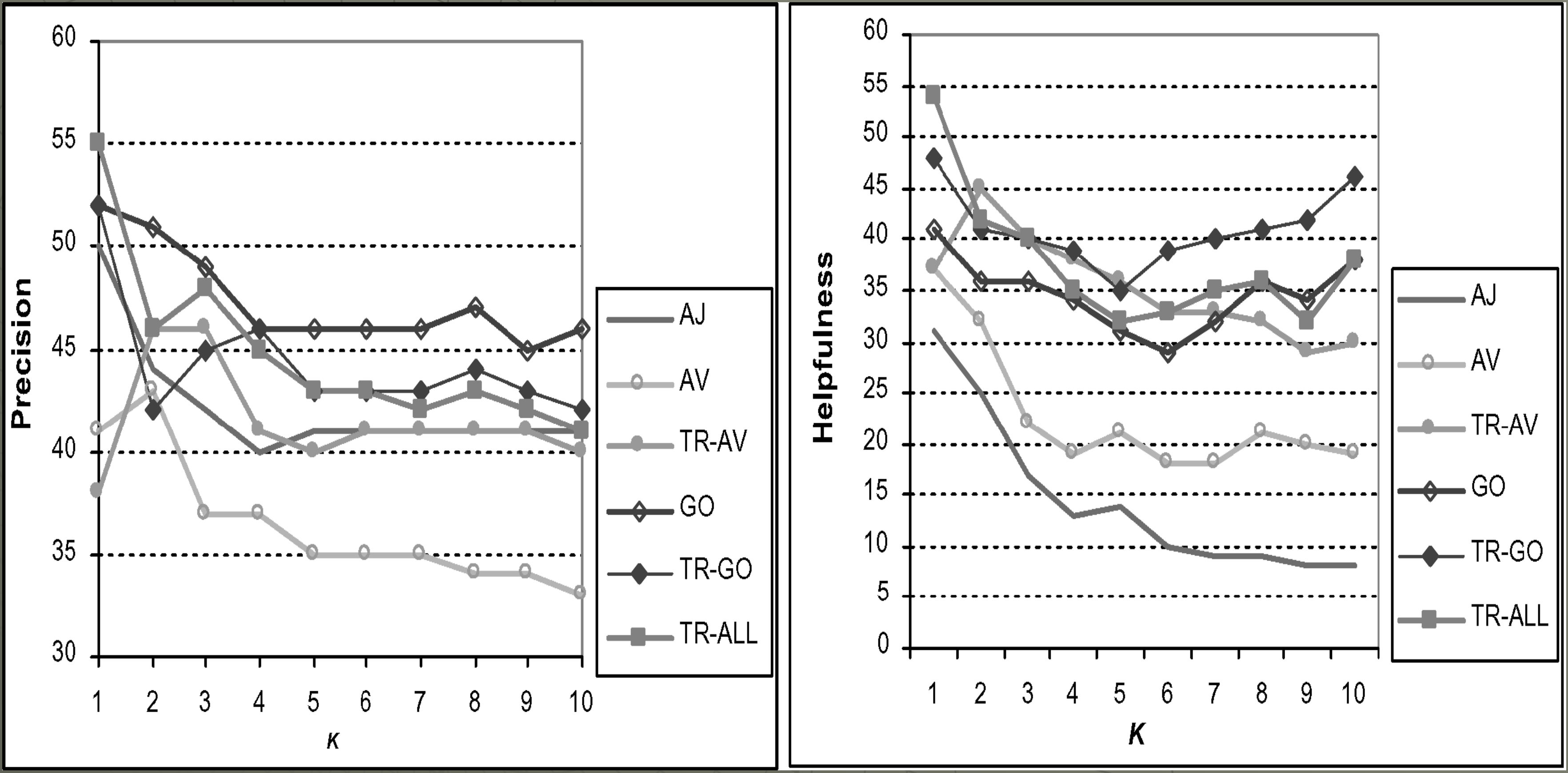
Year 2002 Evaluation



By Colleagues

Evaluation Results

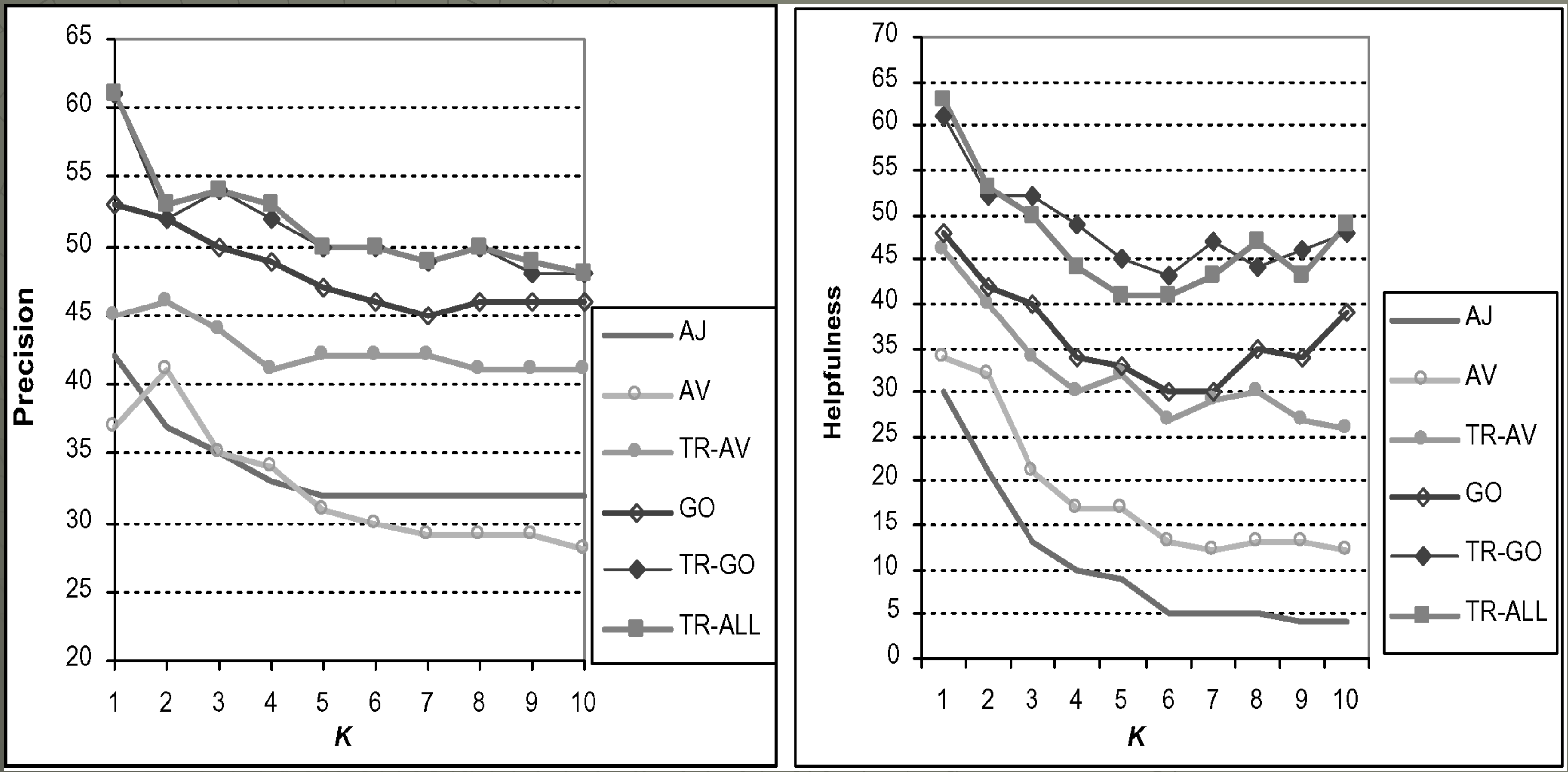
Year 2002 Evaluation



By CiteSeer

Evaluation Results

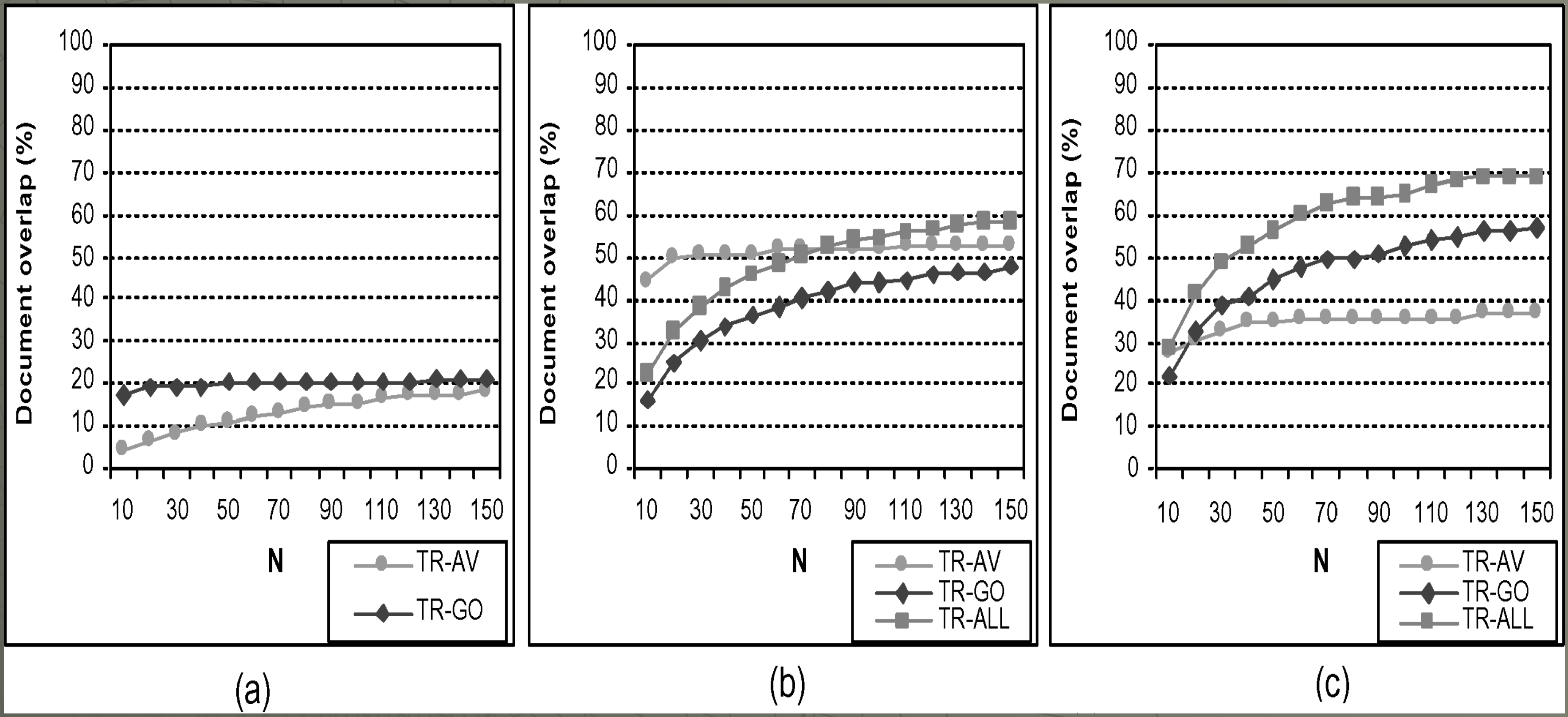
Year 2002 Evaluation



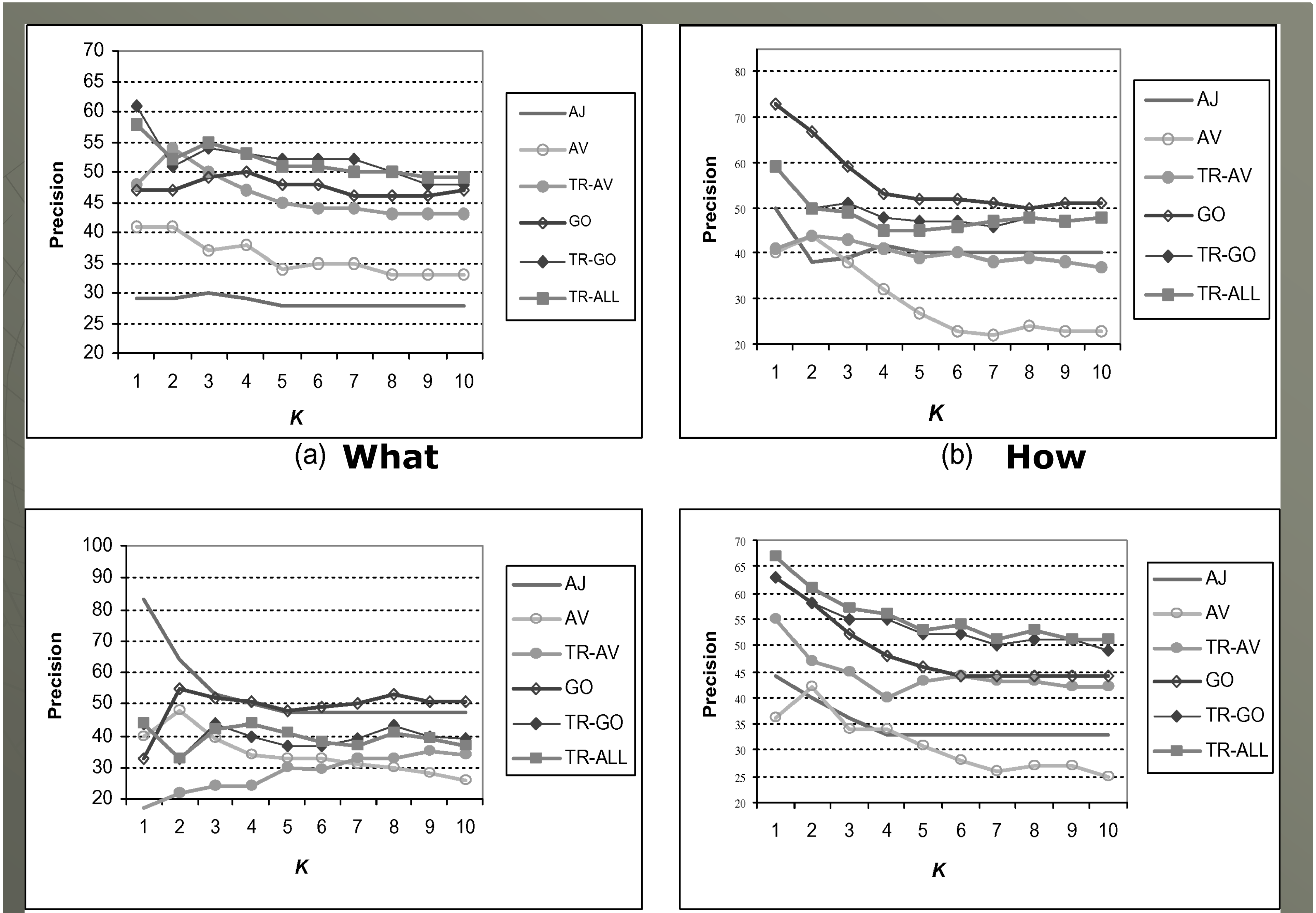
Combined

Evaluation Results

Year 2002 Evaluation



Average percentage of all(a), top 10(b), and relevant in top 10(c) documents contained in top N documents returned for each original query by the underlying search engine during the 2002 evaluation



(a) **What**

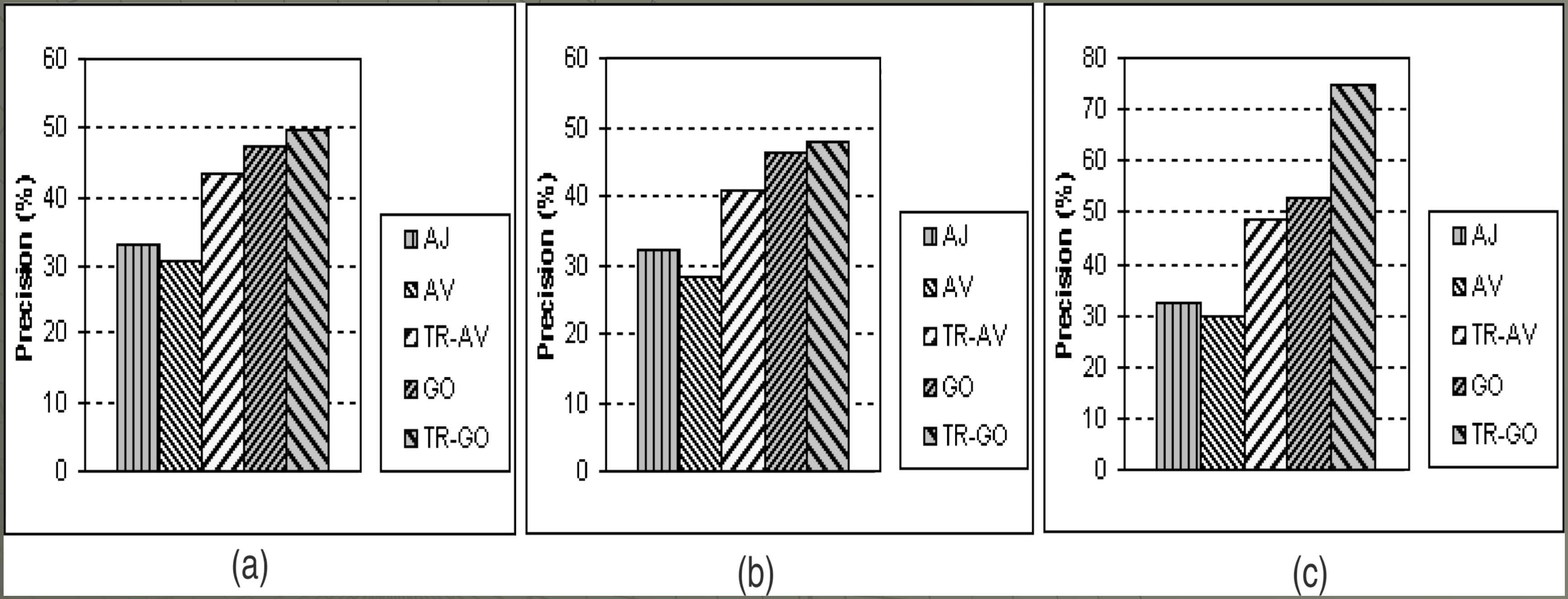
(b) **How**

(c) **Where**

(d) **Who**

Evaluation Results

Year 2002 Evaluation



Average precision for

(a) All 150 documents retrieved

(b) Top 10 documents using current re-ranking

(c) Top 10 documents using perfect re-ranking

Future Work & Summary

Future Work

- Existing methods for document extraction can be implemented.
- Phrase transforms that contain content words from the question.
- Dynamic transformation process.

Summary

A method for learning query transformations that improves the ability to retrieve documents with answers to natural language questions has been introduced.

References

- ROBERTSON, S. http://www.soi.city.ac.uk/~ser/idf.html .
- K. Sparck Jones, S. Walker and S.E. Robertson: A probabilistic model of information retrieval: development and status.
- Eugene Agichtein, Steve Lawrence, Luis Gravano: Learning search engine specific query transformations for question answering. April 2001 Proceedings of the tenth international conference on World Wide Web.
- Susan Dumais, Michele Banko, Eric Bill, Jimmy Lin, Andrew Ng: Web Question Answering System: Is More Always Better?

Comments……