System Summary and Timing
Organization Name: Mayo
List of Run ID's: expst1 expst2

- Construction of Indices, Knowledge Bases, and other Data Structures
  - Methods Used to build Data Structures
    - Length (in words) of the stopword list: 571
    - Manually-Indexed Terms? : CO, IN and GV are used; the given abbreviation of each concept is used as the ID of the concept.
  - Statistics on Data Structures built from TREC Text
    - Inverted index
      - Run ID : expst1
      - Total Storage (in MB): 123 MB
      - Total Computer Time to Build (in hours): 22 CPU hours
      - Automatic Process? (If not, number of manual hours): Yes
      - Only Single Terms Used? : No. The concepts in the CO, IN and GV fields are used, too.
    - Special Routing Structures
      - Run ID : expst1
      - Type of Structure: sparse matrices, similar to inverted-file document indexing
      - Total Storage (in MB): 8 MB (plus 72 MB if counting the WSJ documents which are indexed for the AdHoc Test anyway)
      - Total Computer Time to Build (in hours): 0.13 CPU hours (plus 60 hours if counting the indexing time of the WSJ documents which is needed for the AdHoc Test anyway)
      - Automatic Process? (If not, number of manual hours): Yes
      - Brief Description of Method: We use Expert Network (ExpNet) to expand both routing queries and ad-hoc queries. Given a query, ExpNet identifies its Nearest Neighbors (NNs) among training queries, and then the terms in the documents which are related to the NNs are added to the given query. For the routing task, we set the number of NNs to one. That is, ExpNet behaves just like relevance feedback, and adds to each query the top-ranking terms (1000) in related documents of this query. A weighting factor of 0.45 was experimentally determined to scale the additional terms in the query expansion.
  - Statistics on Data Structures built from TREC Text
    - Inverted index
      - Run ID : expst2
      - Total Storage (in MB): 72 MB
      - Total Computer Time to Build (in hours): 60 CPU hours
      - Automatic Process? (If not, number of manual hours): Yes
      - Only Single Terms Used? : No. The concepts in the CO, IN and GV fields are used, too.
    - Special Routing Structures

- Run ID : expst2
- Type of Structure: sparse matrices, equivalent to inverted-file document indexing
- Total Storage (in MB): 14 MB (plus 72 MB if counting the index of the WSJ documents which are needed anyway)
- Total Computer Time to Build (in hours): 0.3 CPU hour (plus 60 hours if counting the indexing time of the WSJ documents which are needed anyway)
- Automatic Process? (If not, number of manual hours): Yes
- Brief Description of Method: We use Expert Network (ExpNet) to expand both routing queries and ad-hoc queries. Given a query, ExpNet identifies its Nearest Neighbors (NNs) among training queries, and then the terms in the documents which are related to the NNs are added to the given query. For the ad-hoc task, we set the number of NNs to 13. That is, ExpNet chooses 13 NNs (training queries) for each testing query, and adds 1000 top-ranking terms in the related documents to the given query. A weighting factor of 0.10 was experimentally determined to scale the additional terms in the query expansion.

- Query construction
  - Automatically Built Queries (Ad-Hoc)
    - Topic Fields Used: title, desc, narr, con
    - Average Computer Time to Build Query (in cpu seconds): 871/50
    - Method used in Query Construction
      - Term Weighting (weights based on terms in topics)? : Yes. TF*IDF is used, where TF is the with-in-topic-frequency of a term, and the IDF is derived from a document collection (WSJ documents in disk 1 and 2, and SJM documents in disc 3)
      - Heuristic Associations to Add Terms? : Yes, a query-query similarity (cosine value) is used to identify the NNs (13 training queries) of a given query, and the query-document relevance judgments of these NNs are used to expand the given query.
  - Automatically Built Queries (Routing)
    - Topic Fields Used: title, desc, narr, con
    - Average Computer Time to Build Query (in cpu seconds): 524/50
    - Method used in Query Construction
      - Terms Selected From
        I. Topics: yes
        II. Only Documents with Relevance Judgments: yes
    - Term Weighting with Weights Based on terms in
      - Topics: yes
      - Only Documents with Relevance Judgments: yes
    - Heuristic Associations to Add Terms from
      - Topics: yes
      - Only Documents with Relevance Judgments: yes

    * Expansion of Queries using Previously-Constructed Data Structure:
    * Structure Used: sparse matrices (equivalent to inverted files, refer to the report for details).

- Searching
  - Search Times
    * Run ID : expst1
    * Computer Time to Search (Average per Query, in CPU seconds): 438/50
    * Run ID : expst2
    * Computer Time to Search (Average per Query, in CPU seconds): 1290/50
  - Machine Searching Methods
    * Run ID : expst1
    * Vector Space Model? : yes
    * Other: Expert Network (a Nearest Neighbor classification approach) is used in combination with a Vector Space Model.
    * Run ID : expst2
    * Vector Space Model? : yes
    * Other: Expert Network (a Nearest Neighbor classification approach) is used in combination with a Vector Space Model.
  - Factors in Ranking
    * Run ID : expst1
    * Term Frequency? : yes
    * Inverse Document Frequency? : yes
    * Run ID : expst2
    * Term Frequency? : yes
    * Inverse Document Frequency? : yes
  - Machine Information
    * Machine Type for TREC Experiment: SparcStation 10/30
    * Was the Machine Dedicated or Shared: Shared (was also researcher's desktop)
    * Amount of Hard Disk Storage (in MB): 3059 MB
    * Amount of RAM (in MB): 32 MB
    * Clock Rate of CPU (in MHz): 36 MHz
  - System Comparisons
    * Amount of "Software Engineering" which went into the Development of the System: our ExpNet is an research-oriented system; about 4 person-month has been spent on adopting it to the TREC materials and its scale-up
    * Given appropriate resources
      · Could your system run faster? : yes
    * Features the System is Missing that would be beneficial: simplification of data structure and system