Divide and Conquer Strategies for Effective Information Retrieval

Jie Chen and Yousef Saad
Department of Computer Science and Engineering, University of Minnesota

Background

Latent Semantic Indexing (LSI): Let a column vector x_i represent a document. Given a term-document matrix $X = [x_1, \dots, x_n]$ and a query vector q, the relevance scores of the documents to the query are computed (up to normalizations) as the row vector

$$q^T X_k,$$

where X_k is the best rank-k approximation of X.

Limitations: When the document collection (X) becomes large, the computation of X_k is very expensive, both in time and in memory. Let X have size $m \times n$ with nnz nonzeros, the most efficient implementation of truncated SVD takes

- time: $O(k'(nnz + \min(m,n)) + T_t)$,
- space: $O(nnz + k' \min(m,n) + T_s)$,

where k' is the number of Lanczos steps (typically a few times of k), and T_t (T_s) is the time (space) cost of eigendecompositions of tridiagonal matrices and convergence tests.

Our Approach

Two combined schemes to reduce the above costs:

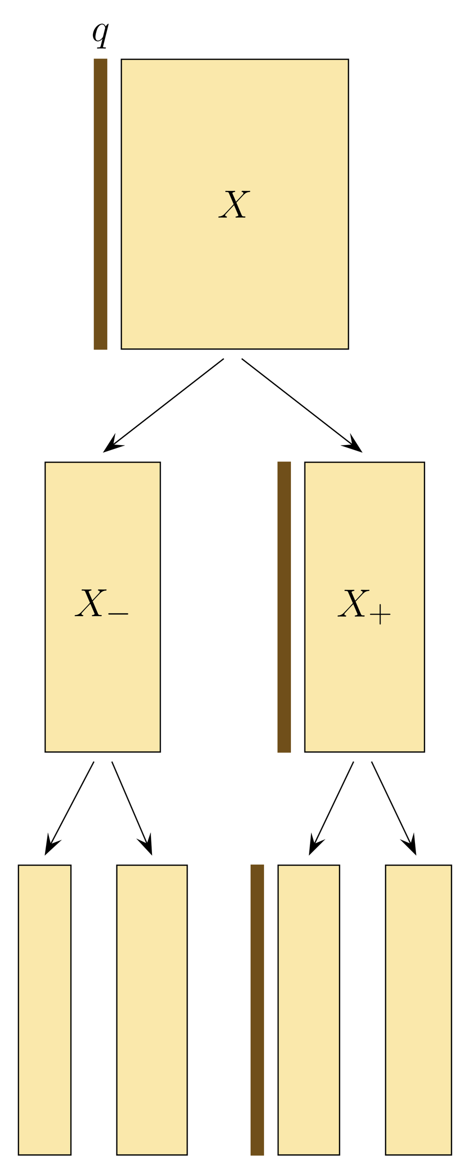
- Partition the term-document matrix, and
- Perform a relevance analysis (similar to LSI) for each partition, or for only a few ones.

Benefits:

1. The partitioning step is simple and very inexpensive.
2. The relevance analysis yields similar results to LSI, but it can be an order of magnitude faster.
3. Each analysis deals with only a small portion of the matrix, hence the whole process is highly parallelizable.
4. Instead of a complicated parallel SVD for large scale LSI, our approach is conceptually much simpler and easier to implement.

Partitioning

Column partitioning (for tall matrix):

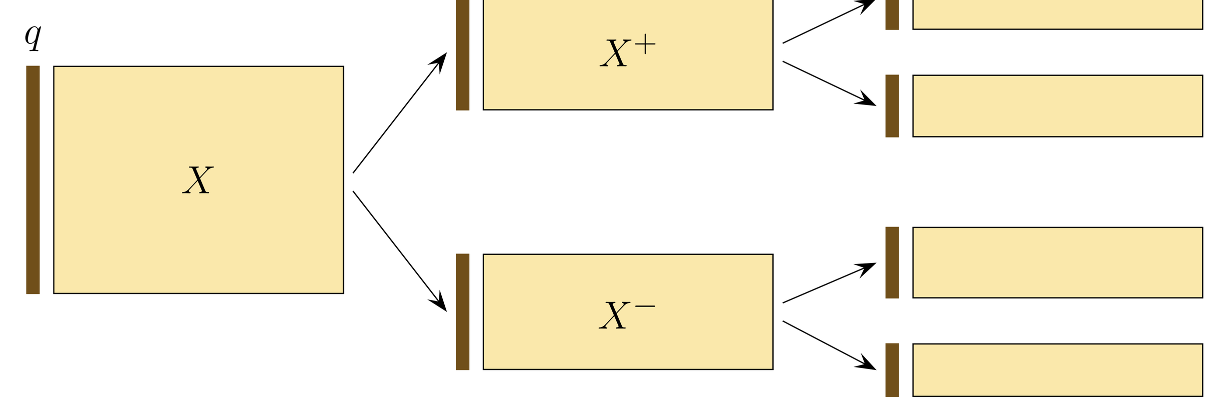


COLUMN-PARTITIONING(X)

1. Compute the centroid c of the documents.
2. Compute the largest right singular vector v of the column-centered matrix $\bar{X} = X - c\mathbf{1}^T$.
3. Form $X_+ \leftarrow \{x_i \mid v_i \geq \text{margin}_-\}$ and $X_- \leftarrow \{x_i \mid v_i < \text{margin}_+\}$.
4. If $|X_+| >$ set.size.threshold, COLUMN-PARTITIONING(X_+).
5. If $|X_-| >$ set.size.threshold, COLUMN-PARTITIONING(X_-).

ROW-PARTITIONING(X)

1. Compute the centroid c' of the terms.
2. Compute the largest left singular vector u of the row-centered matrix $\bar{X}' = X - \mathbf{1}c'$.
3. Form $X^+ \leftarrow \{X(i,:) \mid u_i \geq \text{margin}^-\}$ and $X^- \leftarrow \{X(i,:) \mid u_i < \text{margin}^+\}$.
4. If $|X^+| >$ set.size.threshold, ROW-PARTITIONING(X^+).
5. If $|X^-| >$ set.size.threshold, ROW-PARTITIONING(X^-).



Row partitioning (for wide matrix)

Query Strategies

For tall matrix X:

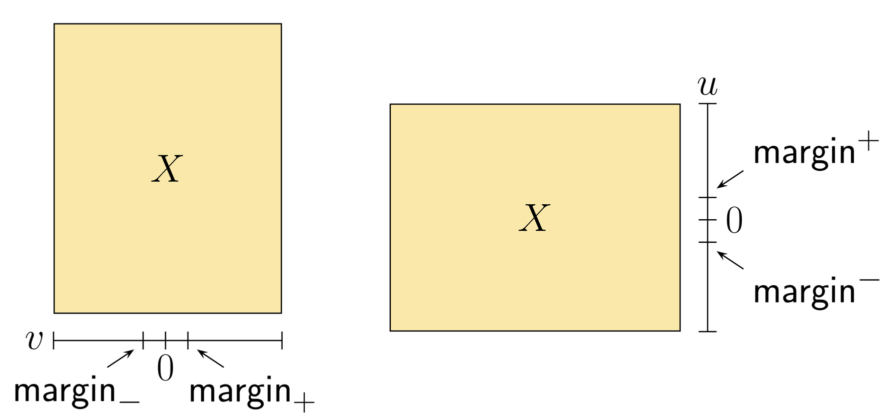
1. Partition the columns of X. A binary tree structure is formed. (See figure on the panel to the left.)
2. For a given query q, compute relevance scores between q and those documents in the leaves of the tree. (See explanation on the panel to the right.)
3. Since a document x_i may belong to different partitions, the relevance score for x_i is the maximum of all the scores computed for x_i in step 2.

For wide matrix X:

1. Partition the rows of X. A binary tree structure is formed. (See figure on the panel to the left.) Denote the resulting leaf nodes $X_{(1)}, X_{(2)}, \dots X_{(p)}$.
2. Partition the query q in the same way.
3. Compute $s' = \sum_{i=1}^{p} q_{(i)}^T \tilde{Q}_k^{(i)} \tilde{Q}_k^{(i)T} X_{(i)}$. (See explanation on the panel to the right.)
4. Scale each entry of s' by the norm of the corresponding column of

$$\begin{bmatrix} \tilde{Q}_k^{(1)} \tilde{Q}_k^{(1)T} X_{(1)} \\ \vdots \\ \tilde{Q}_k^{(p)} \tilde{Q}_k^{(p)T} X_{(p)} \end{bmatrix}.$$

Partitioning (cont.)



Relevance Analysis

How to analyze the relevances between q and X (or a portion of X)?

- In LSI, the relevance scores are computed as the vector $q^T X_k$, modified by scaling each entry with the norm of the corresponding column of X_k.
- Let the SVD of X be $U\Sigma V^T$, then $q^T X_k = q^T (X V_k V_k^T)$ and $q^T X_k = q^T (U_k U_k^T X)$.
- Instead, for tall matrix X, we propose computing the vector $q^T (X Q_k Q_k^T)$ and scaling each entry with the norm of the corresponding column of $X Q_k Q_k^T$, and for wide matrix X, compute the vector $q^T(\tilde{Q}_k \tilde{Q}_k^T X)$ and scale each entry with the norm of the corresponding column of $\tilde{Q}_k \tilde{Q}_k^T X$.
- Here, the columns of Q_k (and $\tilde{Q}_k$) are the Lanczos vectors for the matrix $X^T X$ (and $X X^T$). The major advantage is that they are much less expensive to compute than V_k (and U_k)!

How good is using Lanczos vectors instead of singular vectors?
For any $j < k$,

$$\left[(q^T X) - (q^T X Q_k Q_k^T)\right] u_j \leq c_j T_{k-j}^{-1}(1 + 2\gamma_j),$$
$$\left[(q^T X) - (q^T \tilde{Q}_k \tilde{Q}_k^T X)\right] u_j \leq \tilde{c}_j T_{k-j}^{-1}(1 + 2\gamma_j),$$

where c_j and $\tilde{c}_j$ are some positive constants, γ_j is the eigengap between the j-th and the $(j+1)$-th eigenvalues of $X^T X$, and $T_\ell(x)$ is the Chebyshev polynomial of the first kind of degree ℓ. Fixing x, T_ℓ can be viewed as an exponential function of ℓ:

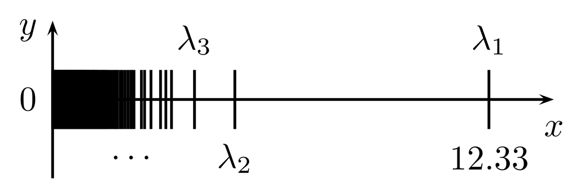
$$T_\ell(x) \approx \frac{1}{2}(\exp(\operatorname{arccosh}(x)))^\ell.$$

Computational costs:

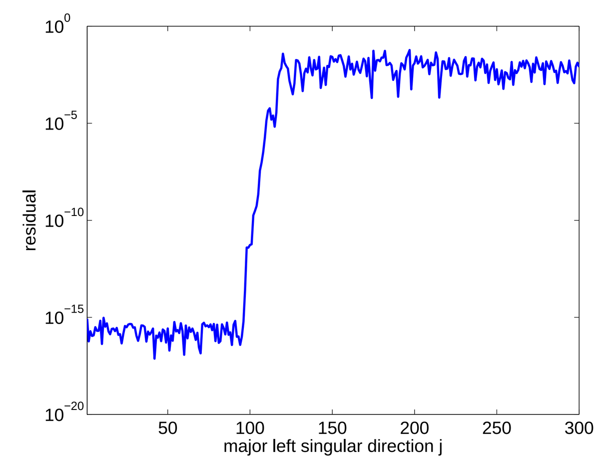
	Singular vectors	Lanczos vectors
Time	$k'(nnz + \min(m,n)) + T_t$	$k(nnz + \min(m,n))$
Space	$nnz + k' \min(m,n) + T_s$	$nnz + k \min(m,n)$

In practice, computing the Lanczos vectors can be an order of magnitude faster than computing the singular vectors.

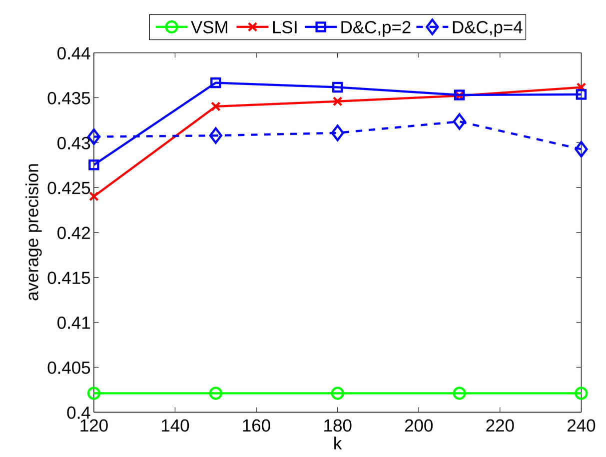
Results



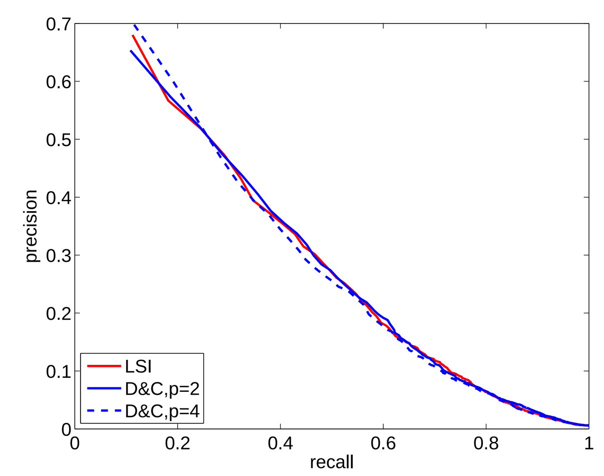
Eigenvalue distribution of the matrix $X^T X$. Note the big eigengaps.



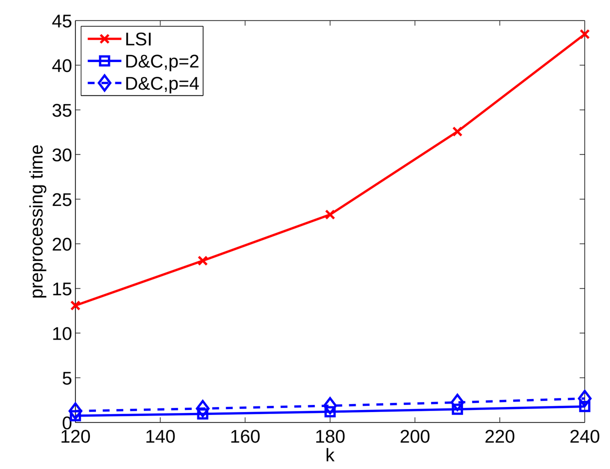
The value of $\left[(q^T X) - (q^T X Q_k Q_k^T)\right] u_j$ for different j (fixing $k = 300$).



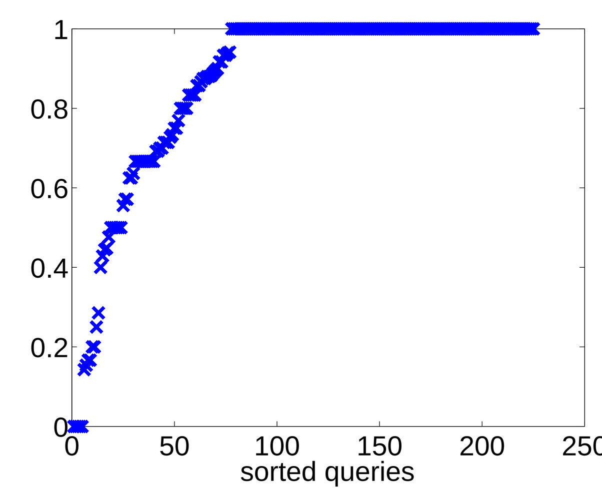
Average query precision for different k.



Precision recall for $k = 200$.



Time cost: our approach v.s. LSI (in seconds).



Percentage of relevant documents that are in the partition where a certain query belongs.

This work was supported by NSF grant DMS 0810938 and by the Minnesota Supercomputing Institute.