Document identifier reassignment through dimensionality reduction

Roi Blanco and Álvaro Barreiro

[1]roi@mail2.udc.es, barreiro@udc.es
AILab, Computer Science Department
University of A Corunna, Spain

27th European Conference on Information Retrieval, 2005

Motivation

- The reassignment of document identifiers is a very recent technique to reduce the size of Inverted Files (IF).
- Inverted Files are the most used indexing structure for Large collections of text.
- We present a solution to the reassignment problem based on reducing the input data dimensionality via Singular Value Decomposition (SVD).

Outline

Inverted Files

Doc.Id	Input Text:
1:	Rain on the green grass
2:	and rain on the tree
3:	And rain on the housetop
4:	but not on me
5:	Rain, rain, go away

term	Documents
rain	<4; 1,2,3,5>
on	<4; 1,2,3,4>
the	<3; 1,2,3>
green	<1; 1>
grass	<1; 1>
and	<2; 2,3>
tree	<1; 2>
housetop	<1; 3>
but	<1;4>
not	<1;4>
me	<1;4>
go	<1;5>
away	<1;5>

- An IF consists on a *dictionary* and a *concordance* file.
- The concordance file is a set of posting lists
- Each posting list follows the format
  - $< t_i; f_{t_i}; d_1, d_2, \ldots, d_{f t_i} >, d_i < d_j \forall i < j$
  - f_{t_i} stands for the frequency of the term t_i (number of documents in which t_i appears)
  - d_i is the document identifier
- Also, a posting list can store additional information
  - A sequence of numbers giving the exact position of a term in the text

Coding methods

- Posting lists usually are compressed
- Actually, posting lists store the gaps between documents, which are codified.
- Compression methods exploit the fact that small d-gaps occurs often, giving short codes to them.
- Compression saves file space, query access time (improves I/O and seeking times).

Document Identifier Reassignment

- An inverted file can be seen as a posting list set containing a sequence of encoded d-gaps
- The document reassignment problem tries to find the bijective function f that
  - maps each document identifier into a new identifier in the range $[1 \ldots d]$
  - minimizes the bits used for coding the d-gaps.
- In general is a hard task
- No completely satisfying solution proposed up to date

Previous work

- Different works addressed the problem from different perspectives:
  - Enhancing locality in inverted files.
  - Minimizing the distance between consecutive documents.
  - Clustering while indexing.
- Different approaches to the problem, require different representations of the data.

- Some works build a *weighted similarity graph G*
  - Nodes v_i, v_j represent the document identifiers i, j
  - An edge (v_i, v_j) represents the similarity between documents i and j
- Blandford and Blelloch, IEEE DCC'02 proposed an algorithm that:
  - recursively splits G into small subgraphs $G_{l,i} = (V_{l,i}, E_{l,i})$ until every subgraph becomes a singleton.
  - Reorders the document identifiers, by *depth-first* traversal.

  - Real efficiency is dependant on the collection through graph sparseness (induced by the τ parameter).
  - Execution times reported only for 32.000 documents
  - Needs parameter tuning (τ, graph sparseness, ρ sampling for splitting the graph).

- Shie et al. IP&M 2003, modeled the problem as a *Travelling Salesman Problem* (TSP)
- The TSP is an strategy for the Doc.Id. Reassignment Problem.
  - Given $G = (V, E)$ where $e(v_i, v_j)$ is the weight for the edge from v_i to v_j and V is the set of documents, find a minimal path $P = \{v_1, v_2, \ldots, v_n\}$ containing all the vertexes in V, such as if $P' = \{v'_1, v'_2, \ldots, v'_n\}$ is another path in G, $\sum_{i=2}^{n} e(v_i, v_{i-1}) \leq \sum_{i=2}^{n} e(v'_i, v'_{i-1})$
  - The weight $e(v_i, v_j)$ is the complement of the similarity between documents i and j.

- They apply some heuristic-based solutions (spanning tree, greedy).
- Computes a $d \times d$ *similarity matrix*
- Requires matrix partition techniques for large graphs
- Best results obtained with the Greedy-NN
- For a 475 MB Collection, in [1] is reported that a Greedy algorithm approximating the TSP solution
  - Takes more than 23 hours
  - Space required is 5 times the size of the collection.

- Silvestri et al. ACM SIGIR'04 proposed a different approach
  - Uses a compact representation of the documents, by MD5 hashing the terms
  - Operates *assigning* the document identifiers *on the fly* during the inversion of the text collection.
  - Uses Bottom-Up and Top-Down clustering strategies.
  - Good results in compression and efficiency for the *Google Programming Contest collection*.
  - Space usage is dependant on the average document size.
- Our approach solves the problem as a TSP as Shie et Al, IP&M 2003, but
  - Solves the problem in a reduced dimensionality space,
  - improving the efficiency of the approach.

SVD

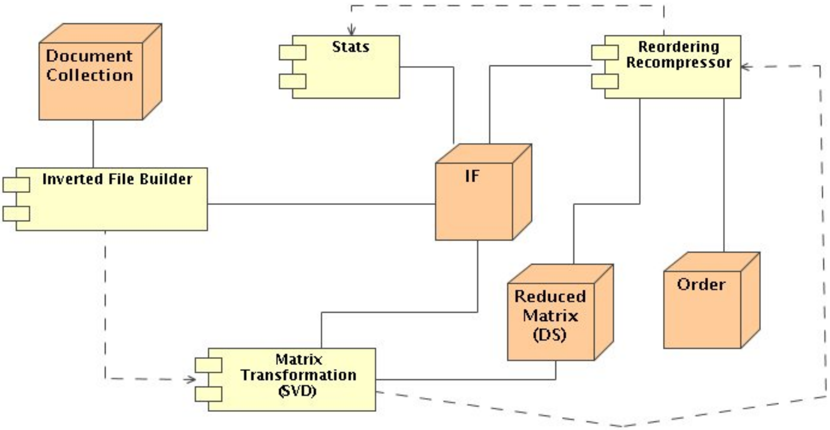
- Modified Greedy approach to the TSP problem:
  - Reduces the dimensionality of the input matrix applying Singular Value Decomposition (SVD) to the Inverted File
  - Avoids the construction of the $d \times d$ similarity matrix
- The original IF is decomposed in $X_{t \times d} = T_{0_{t \times m}} S_{0_{m \times m}} D'_{0_{m \times d}}$ and k-ranked as $X \approx \hat{X}_{t \times d} = T_{t \times k} S_{k \times k} D'_{k \times d}$
  - $\hat{X}$ is the approximation of X obtained with the highest k eigenvalues.

- Theoretical support:
  - $\hat{X}$ is the closest rank k approximation of X in terms of the Euclidean or Frobenious norms.
  - It is the matrix which minimizes $||X - \hat{X}||_N^2$
- The similarity matrix $\Theta(X)$ is k-approximated by $\Theta(X) \approx \Theta(\hat{X}) = \hat{X}'\hat{X} = DS^2D'$
- It is also the closest rank k approximation
- If $D_{d \times k} = \{z_{ij}\}$ and $\{s_i\}$ is the set of diagonal elements of S, it is easy to prove that $\Theta(\hat{X})_{ij} = \sum_{\gamma=0}^{k-1} z_{i\gamma} z_{j\gamma} s_\gamma^2$ avoiding storing the full $\Theta(X)$ matrix

- Therefore:
  - It is possible to calculate $\Theta(\hat{X})$ only storing the k elements of matrix S and the $d \times k$ matrix D.
  - Due to the uniqueness of SVD, the best rank k approximation of $\Theta(X)$ is obtained without computing the full similarity matrix,
  - we just need the inverted file X.

- Applying a Greedy approximation to the TSP in the reduced space:
  - allows a controlled memory usage
  - gives consistent results through different document and collection sizes and heuristics
  - gives results in the order of the obtained by working with the full matrix.

Indexing, Reordering, Recompressing



Evaluation

- We used the TREC-5 LATimes and FBIS Collections
- No stemming, every word is indexed, binary matrix X.

Collection	FBIS	LATimes
Size of the Collection	470 MB	475 MB
Number of distinct terms	209,782	167,805
Number of distinct documents	130,471	131,896

Coding methods tested

- Global no parametric coding:
  - Unary Code
  - Gamma Code
  - Delta Code
- Local codings:
  - Local Golomb
  - Interpolative Code

Software and Hardware

- Indexing and Recompressing tasks: MG4J (University of Milan).
- For computing the SVD: SVDLIBC (based on the SVDPACKC library)
- Modified the previous modules and wrote the reordering/recompressing software in Java.
- Hardware: AMD 2.5Ghz, 1 GB RAM, 40 GB HDD..

- Bits per document gap for the Greedy-NN TSP approach under reduced dimensionality ($k = 200$).
- LATIMES

	Random	Original	Reordered	% Rnd	% Ori
Gamma	8.15	7.77	6.71	17.7	13.48
Delta	7.65	7.25	6.29	17.8	13.3
Interp	6.08	5.88	5.25	13.7	10.8

- FBIS

	Random	Original	Reordered	% Rnd	% Ori
Gamma	7.84	6.74	6.20	21	8.1
Delta	7.35	6.35	5.80	21.1	8.7
Interp	5.83	5.25	4.98	14.6	5.2

- As expected improvements are better respect to randomized collections.
- As expected, worse improvements were achieved with Interpolative Coding (lower original values).
- Good overall performance results.
- Total running time(elapsed) about 8 hours (indexing, reordering and recompressing).

c-Greedy-NN

- c-Greedy-NN: new algorithm designed to exploit the dimensionality reduction in a straight way
  - Based on the division of the original problem in c subproblems
  - Changes only a minimum and isolated part of the software
  - It divides the DS matrix in c blocks of $[d/c]$ documents
  - Each block is reordered by running the greedy algorithm.
  - A block order is decided by running another greedy with c documents each one selected from different blocks

c-Greedy-NN Results

- $k = 200$, delta coding.

	c (LATimes)					
	70	100	200	300	500	1000
BPG	6.68	6.72	6.81	6.87	6.95	7.03
Time	18m8s	9m50s	3m21s	1m57s	1m7s	42s

	c (FBIS)					
	70	100	200	300	500	1000
BPG	5.98	6.00	6.05	6.09	6.14	6.22
Time	17m37s	9m35s	3m15s	1m53s	1m5s	40s

- Running times are as expected from the analytical form.

Final Remarks

- Reassignment problem solved as a TSP in a reduced dimensionality space for efficiency purposes.
- Improvements in compression ratios, efficient coding with static methods.
- Good compression ratios with acceptable running times can be achieved.

- Further Research
  - Test with other heuristics for the TSP in the reduced dimensionality space
  - Testing other techniques (for example clustering) in the reduced dimensionality space.
  - Try alternative more efficient implementations of SVD and/or other techniques for matrix reduction.
  - Test with web collections.

For Further Reading I

- W.-Y. Shieh, T.-F. Chen, J. J.-J. Shann and C.-P. Chung.
  Inverted file compression through document identifier reassignment.
  Information Processing and Management 39(1):117-131, January 2003.

- D. Blandford and G.Blelloch.
  Index compression through document reordering.
  Proceedings of the IEEE Data Compression Conference (DCC'02), pp. 342-351, 2002.

For Further Reading II

- F. Silvestri, S. Orlando and R. Perego.
  Assigning identifiers to documents to enhance the clustering property of fulltext indexes.
  Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 305-312, 2004.

- I. H. Witten, A. Moffat and T. C. Bell. *Managing Gigabytes - Compressing and Indexing Documents and Images*, 2nd edition. Morgan Kaufmann Publishing, San Francisco, 1999.