Experiments with LSA Scoring: Optimal Rank and Basis

John Caron[*]

1 Introduction

Latent Semantic Analysis (LSA) is one of the main variants of vector space methods for information retrieval, and continues to be an active area of research, both in the theory of how LSA works and in the practical applications of the method. Alternatives to the Singular Value Decomposition (SVD) have been explored that offer improvements in storage, speed, updating or other advantages, especially for large datasets. Alternatives to constructing the term-document matrix, such as term-weighting schemes have also been explored.

At the core of LSA is the scoring of a query against a canonical set of documents, using the inner product of their vector representations. Virtually all LSA researchers have assumed a particular scoring function, and alternatives have not been well investigated. In this paper I define a family of scoring functions parameterized by a "weighting exponent" p, which weights the score by Σ^p, where Σ is the diagonal matrix of SVD eigenvalues. I present empirical findings on how retrieval accuracy varies as a function of p and the SVD approximation rank k, and find that often the standard scoring function, corresponding to $p = 0$, is not optimal.

In Section 2, I review the Singular Value Decomposition, and define LSA scoring functions as a choice of basis for the vector space defined by the document set. In Section 3, I test these scoring functions on standard test datasets, for different values of the SVD approximation rank k. In Section 4, I present results from a similar experiment on matching question with answers, within the context of the Frequently Asked Question Organizer (FAQO), a prototype system for technical support that I developed. In Section 5, I summarize related work, and discuss the significance of these results.

[*]University Corporation for Atmospheric Research, and Computer Science Dept. University of Colorado at Boulder

2 SVD bases for LSA scoring

Singular Value Decomposition

Any real matrix A can be decomposed (assume $n < m$):

$$A^{(mxn)} = U^{(mxn)}\Sigma^{(nxn)}V^{T(nxn)}$$

into matrices U, Σ, and V, where $U^TU = V^TV = I_n$ are column orthonormal, and Σ is diagonal with elements $(\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq \ldots \geq \sigma_n \geq 0)$, called the*singular values* of A. The number of non-zero singular values is the *rank* of A.

The first k column vectors of U, Σ, and V can be used to form the matrix

$$A_k^{(mxn)} = U_k^{(mxk)}\Sigma_k^{(kxk)}V_k^{T(kxn)}$$

which has the important property that it is the best rank-k approximation to A: $\|A^k - A\|_F$ is minimum for all rank-k matrices (Frobenius norm). The matrix $A_k = U_k\Sigma_kV_k^T$ is the *SVD rank-k approximation* to A.

The Term - Document Matrix

Starting with a set of n documents called the *canonical document set*, we extract a set of m terms called the *canonical term set*. We then form the *mxn term-document matrix* A whose elements a_{ij} are some function of the frequency of the ith term in the jth document.

Any document can be made into a document vector of length m by setting the ith element to some non-zero value if the document contains the corresponding canonical term, and to zero otherwise. If a document contains none of the canonical terms, then its vector is the null vector. Clearly the set of canonical terms determine how and if a document can be represented effectively. Let D be the set of all possible document vectors formed using the canonical term set, and for convenience we will assume that $D = R^m$.

The columns of $A = \{\vec{d_j}, j = 1..n\}$ are the document vectors for the canonical document set. The set of all linear combinations of these vectors defines a vector space $F = Span(A) \subset R^n$. The projection of an arbitrary document vector $\vec{q} \in D$ into F can be found by forming the inner product of $\vec{q}$ with each $\vec{d_j}$. In matrix form: $\vec{q}_{proj} = A^T \cdot \vec{q}$. So A^T is a linear transformation from R^m to R^n, $A^T : D \rightarrow F$, which takes a document vector $\vec{q} \in D$ and represents it as a linear combination of the canonical document vectors $\{\vec{d_j}\}$.

Basis sets for Span(A)

The SVD decomposition provides a basis set for $F = Span(A)$, namely the columns of U. To see this, we have to show that any $\vec{d_j}$ can be expressed as a linear combination of the columns of U. If $\vec{e}_j$ is the jth unit column vector, $\{\vec{u}_j\}$ are the column vectors of U and v_{ij} are the matrix elements of V, then

$$\vec{d_j} = A \cdot \vec{e}_j$$

$$= U\Sigma V^T \cdot \vec{e}_j$$
$$= \sigma_1 v_{j1} \vec{u}_1 + \sigma_2 v_{j2} \vec{u}_2 + \ldots + \sigma_n v_{jn} \vec{u}_n$$

Since the $\{\vec{u}_j\}$ are orthonormal ($U^T U = I \Rightarrow \vec{u}_i \cdot \vec{u}_j = \delta_{ij}$) then the $\{\vec{u}_j\}$ are an orthonormal basis set for F.

There are other possible basis sets for F that fall out of the SVD decomposition. For example (anticipating the next sections) for any integer b, $U\Sigma^b$ is a basis for F, since each $\vec{d}_j$ is a linear combination of the columns of $U\Sigma^b$:

$$\vec{d}_j = \sigma_1^{1-b} v_{j1} (\vec{u}_1 \sigma_1^b) + \sigma_2^{1-b} v_{j2} (\vec{u}_2 \sigma_2^b) + \ldots + \sigma_n^{1-b} v_{jn} (\vec{u}_n \sigma_n^b)$$

Since the columns of $U\Sigma^b$ are scalars times the columns of U, they are orthogonal (although not normalized).

LSA Document Retrieval Scoring

Generally we are interested in retrieving documents from the canonical document set $\{\vec{d}_j\}$ that are similar to some document $\vec{q}$ not in the canonical set. To do so we compute a score based on a comparison of $\vec{q}$ with each $\vec{d}_j$, then return the $\{\vec{d}_j\}$ sorted by highest score. The standard scoring algorithm uses the cosine between two vectors: $cos(\vec{a}, \vec{c}) = \vec{a}^T \cdot \vec{c} / \|\vec{a}\|_2 \|\vec{c}\|_2$, and in what follows I explore ways to choose the vectors representing $\vec{q}$ and $\vec{d}_j$ in calculating the score.

(1) A straightforward approach is to use the columns of $A_k = A_k \vec{e}_j$ as approximations to the document vectors $\vec{d}_j$ and directly take the cosine with $\vec{q}$, so that the score of the jth document against $\vec{q}$ is:

$$score_j = cos(A_k \vec{e}_j, \vec{q}) = \frac{\vec{e}_j^T V_k \Sigma_k U_k^T \cdot \vec{q}}{\|\vec{e}_j^T V_k \Sigma_k U_k^T\|_2 \|\vec{q}\|_2} = \frac{\vec{e}_j^T V_k \Sigma_k U_k^T \cdot \vec{q}}{\|\vec{e}_j^T V_k \Sigma_k\|_2 \|\vec{q}\|_2} \qquad (1)$$

(2) Another approach is to observe that

$$A \approx A_k = U_k \Sigma_k V_k^T \Rightarrow U_k^T A \approx \Sigma_k V_k^T$$

implies that the columns of $\Sigma_k V_k^T$ approximate the $\vec{d}_j$ in the vector space that has U_k as its basis set. The query projected to that basis is $U_k^T \vec{q}$, so:

$$score_j = cos(\Sigma_k V_k^T \vec{e}_j, U_k^T \vec{q}) = \frac{\vec{e}_j^T V_k \Sigma_k \cdot U_k^T \vec{q}}{\|\vec{e}_j^T V_k \Sigma_k\|_2 \|U_k^T \vec{q}\|_2} \qquad (2)$$

$$score_j = (\sigma_1 v_{j1} \vec{u}_1 \cdot \vec{q} + \sigma_2 v_{j2} \vec{u}_2 \cdot \vec{q} + \ldots + \sigma_k v_{jk} \vec{u}_k \cdot \vec{q}) / norms$$

which gives the same retrieval results as (1) because the $\|\vec{q}\|_2$ in (1) and the $\|U_k^T \vec{q}\|_2$ in (2) are constant for all scores, and so do not affect the document rankings.

(3) Similarly:

$$A \approx U_k \Sigma_k V_k^T \Rightarrow \Sigma_k^{-1} U_k^T A \approx V_k^T$$

implies that the columns of V_k^T approximate the $\vec{d_j}$ in the vector space that has $U_k \Sigma_k^{-1}$ as its basis set. The query in that basis is $\Sigma_k^{-1} U_k^T \vec{q}$, so:

$$score_j = cos(V_k^T \vec{e}_j, \Sigma_k^{-1} U_k^T \vec{q}) = \frac{\vec{e}_j^T V_k \cdot \Sigma_k^{-1} U_k^T \vec{q}}{\|V_k^T \vec{e}_j\|_2 \|\Sigma_k^{-1} U_k^T \vec{q}\|_2} \tag{3}$$

$$score_j = (\sigma_1^{-1} v_{j1} \vec{u}_1 \cdot \vec{q} + \sigma_2^{-1} v_{j2} \vec{u}_2 \cdot \vec{q} + \ldots + \sigma_k^{-1} v_{jk} \vec{u}_k \cdot \vec{q})/norms$$

(4) More generally, for an integer p:

$$A \approx U_k \Sigma_k^{-p/2} \Sigma_k^{1+p/2} V_k^T \Rightarrow \Sigma_k^{p/2} U_k^T A \approx \Sigma_k^{1+p/2} V_k^T$$

implies that the columns of $\Sigma_k^{1+p/2} V_k^T$ approximate the $\vec{d_j}$ in the vector space that has $U_k \Sigma_k^{p/2}$ as its basis set. The query in that basis is $\Sigma_k^{p/2} U_k^T \vec{q}$, so:

$$score_j = cos(\Sigma_k^{1+p/2} V_k^T \vec{e}_j, \Sigma_k^{p/2} U_k^T \vec{q}) = \frac{\vec{e}_j^T V_k \Sigma_k^{1+p/2} \cdot \Sigma_k^{p/2} U_k^T \vec{q}}{\|\Sigma_k^{1+p/2} V_k^T \vec{e}_j\|_2 \|\Sigma_k^{p/2} U_k^T \vec{q}\|_2} \tag{4}$$

$$score_j = (\sigma_1^{1+p} v_{j1} \vec{u}_1 \cdot \vec{q} + \sigma_2^{1+p} v_{j2} \vec{u}_2 \cdot \vec{q} + \ldots + \sigma_k^{1+p} v_{jk} \vec{u}_k \cdot \vec{q})/norms$$

Using $U_k \Sigma_k^{p/2}$ as basis for F is equivalent to scaling each basis vectors $\vec{u_i}$ by $\sigma_i^{p/2}, i = 1 \ldots k$. Since both $\vec{q}$ and the $\{\vec{d_j}\}$ are projected into F, the effect on $score_j$ is to 1) weight the contribution of the ith eigenvector by an additional factor of σ_i^p, and 2) divide the score by the norm of the jth column vector of $\Sigma_k^{1+p/2} V_k^T$. (Note that the constant factor $\|\Sigma_k^{p/2} U_k^T \vec{q}\|_2$ does not affect the retrieval results). This defines a family of LSA scoring functions parameterized by p, which might be called the *LSA score-weighting exponent*. Note that (2) and (3) are the special cases where $p = 0$ and $p = -2$, respectively.

Almost all LSA researchers use (1) or (2) as the scoring function (also see Section 5), but I have not been able to find a theoretical basis for that assumption. At the heart of LSA is the dimension reduction of the SVD rank-k approximation, which (somewhat non-obviously) provides a better solution for the problems of polysemy and synonymy than representing the document and query vectors in exact form [4], as for example in the vector space model [7]. In this context, the optimal approximation of the documents and queries for the purpose of scoring is unclear. A future theory of how LSA works should hopefully clarify this, and will likely involve some characterization of the statistical correlation between document sets and queries. In the meanwhile, empirical results may be useful. In the next section, I explore the effect of varying the LSA score-weighting exponent on retrieval accuracy on several standard test datasets.

3 Effect of the weighting exponent on retrieval

To examine the effect of the stretching parameter on retrieval accuracy, I created a Java program for constructing the term-document matrix, and used the SVDPACKC software (las2.c) for the SVD solver [10]. This environment was developed as part of the FAQO project, described in the next section and at [3].

Table 1. *Characteristics of test datasets*

	Medline	Cranfield	CISI	Time
# docs	1033	1400	1460	425
# terms	5696	4516	5483	10595
avg.#terms/doc	50	58	46	199
% non-zero	.87%	1.3%	.85%	1.9%
# queries	30	225	112	83

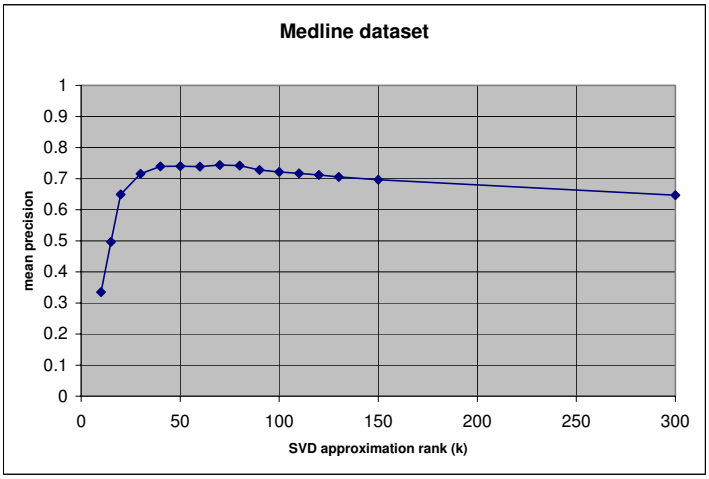

Figure 1. *Standard scoring algorithm (Medline dataset). Performance vs. number of dimensions (compare to Figure 1 of [6]).*

I reproduced as closely as possible the results of Susan Dumais' early work on LSA [6], by using the same term weighting algorithm ("Log(TF) Entropy"), and a similar procedure for constructing the canonical term list. I also normalized the columns of the term-document matrix $\boldsymbol{A}$, which she may not have.

I also used the same test datasets (available from the SMART ftp site [9]) as she did, and Table 1 shows the characteristics of my document matrices, which can be compared to her Table 1. Figure 1 shows the results of varying the number of dimensions (k), using the standard scoring algorithm (corresponding to p=0), on the Medline dataset. This can also be directly compared to her Figure 1 in [6]. Mean precision is the precision averaged over recall levels of 25%, 50% and 75%. Note that in my Figure 1 it is clearer that the LSA algorithm performs almost identically for this dataset for a range of dimensions between k=40 (mean precision =.7391) and k=90 (.7276) with a maximum around k=70 (.7435). This figure shows that my experimental setup gives very similar results to that of other researchers.

In Figures 2, 3, 4, and 5 I show the mean precision of the retrieval as a function of the weighting exponent, for various values of k (the SVD approximation rank),

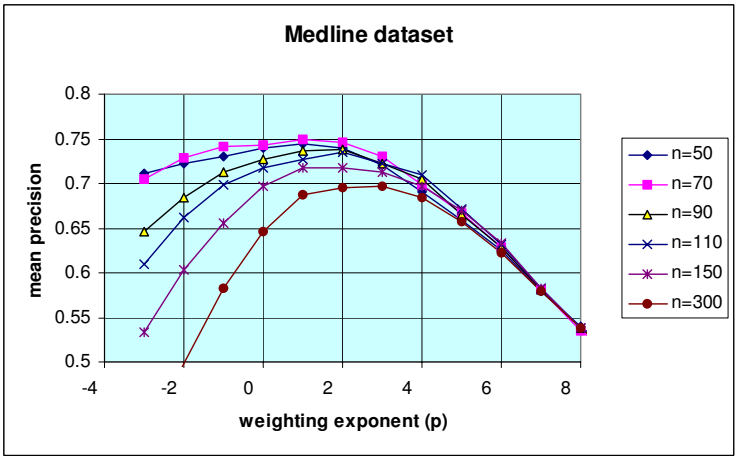

Figure 2. *Medline dataset: Retrieval accuracy vs. p (weighting exponent) for various values of k (SVD approximation rank)*

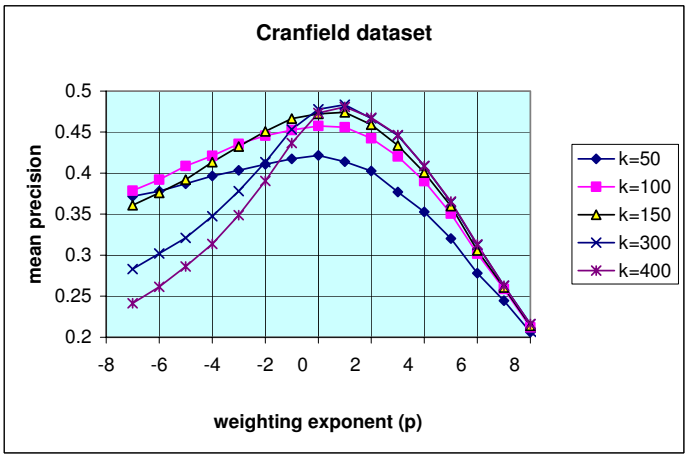

Figure 3. *Cranfield dataset: Retrieval accuracy vs. p (weighting exponent) for various values of k (SVD approximation rank)*

for the Medline, Cranfield, Time and CISI datasets respectively. Some comments on these figures:

• These curves are continuous for the most part. While the difference between most of these numbers is relatively small, it does not appear due to randomness. Figure 6 shows a higher resolution plot of the Medline dataset near the optimal k=70. Here some scatter is seen, probably due to the small number of queries, but the shape of the curves as a whole is not random.

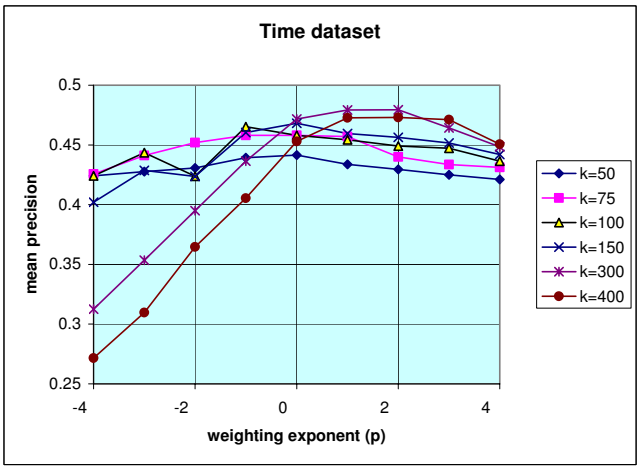

Figure 4. *Time dataset: Retrieval accuracy vs. p (weighting exponent) for various values of k (SVD approximation rank)*

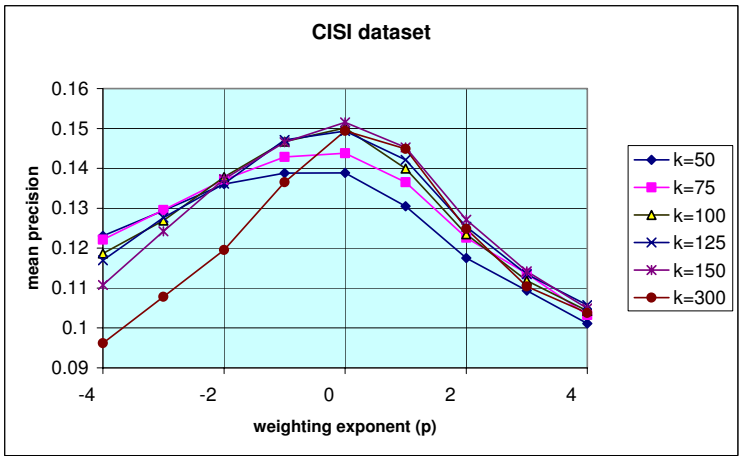

Figure 5. *CISI dataset: Retrieval accuracy vs. p (weighting exponent) for various values of k (SVD approximation rank)*

• All of the curves for different values of k converge as p increases towards the right side of the plots. This is most clear in Figures 2 and 3, which show retrieval results for values of p up to 8. This is further discussed in section 5.

• The optimal value of p depends on k, and on the dataset, and generally increases as k increases. For the limited range of k tested here, the optimal value of p ranges between -1 and 2.

• For the Medline, Cranfield, and Time datasets (Figures 2, 3, 4, 6), at the

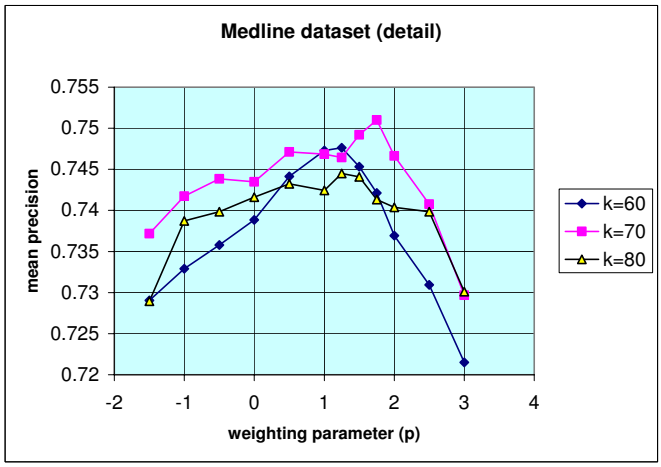

Figure 6. *Medline dataset (detail): Retrieval accuracy vs. p (weighting exponent) for values of k near the optimum.*

optimal value of k, the optimal value of p is around 1.75, 1, and 2 respectively. These are respectively 1.0%, 1.2% and 1.7% improvements over the standard basis (p=0). For non-optimal values of k, the optimal value of p can do quite a bit better than the standard basis, e.g. for the Medline dataset and k=300, the optimal basis is about 7% better than the standard basis.

• For the CISI dataset, the standard basis (p=0) is optimal for all values of k tested. At the optimal value of k=150, the standard basis is about 4% better than p=1. It may be noteworthy that the LSA algorithm has the lowest mean precision for the CISI dataset, 3-5 times lower than the other datasets.

4 Experimental results with the Frequently Asked Question Organizer (FAQO)

The Frequently Asked Question Organizer (FAQO) is a prototype system to assist technical support personnel in answering customer questions [3]. It uses Latent Semantic Analysis for matching queries against a database of previously asked questions and answers. It is currently under active development and testing for technical support at Unidata/UCAR.

To evaluate the usefulness of FAQO, the technical support person rates the documents returned. However s/he typically only rates the first 10 or perhaps 20 documents, rather than systematically finding all documents relevant to a query. Therefore, it is not possible to calculate traditional measures such as recall using these ratings.

As an alternative, I developed a simple methodology called "question/answer matching" for comparing LSA scoring algorithms. The document database consists

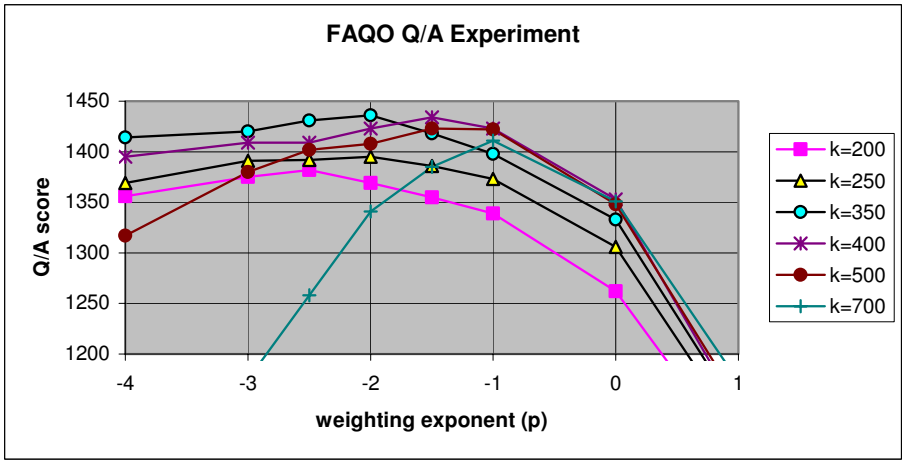

Figure 7. *FAQO QA Experiment: Total Q/A matching score vs. p (weighting exponent) for various values of k (SVD approximation rank)*

of email messages that often contain a single, easily identified question and answer. A set of these is manually identified and the question and answer are separately tagged. The LSA matrix is generated as usual, except that for the tagged messages, only the answer text is used in the document vector. Each question is then used as a query, and the top 10 scoring documents are returned. For each question, a score of 10 is given if the answer is the first returned document, a score of 9 for a rank-two answer, and so on, down to a score of one for a rank 10 answer. Answers that have rank higher than 10 get no points. The algorithm's overall score is then the sum of these scores.

While this algorithm Q/A score is not as good as traditional recall and precision metrics, the score is simple to do, it plausibly measures the usefulness of the results for a user, and it is not obviously biased towards any particular algorithm.

Figure 7 shows the results of this experiment using one set of emails gleaned from the Java Advanced Imaging mailgroup. This dataset contained a total of 979 emails, out of which 209 emails with clear questions and answers were identified and used as queries. The maximum possible score for the test is 2090.

Unlike the tests using the standard datasets in section 3, here values of p less than 0 were optimal. At the optimal value of k=350, the optimal value of p was -2, which is 7.7% better than the standard basis of p=0. For various values of k, the optimal value of p varied from -2.5 to -1, and again increased as k increased. A weighting exponent of -2 is equivalent to dividing the contribution of the ith eigenvector by a factor of σ_i^2.

In concrete terms, the difference between using the standard basis of p=0 and the optimal basis of p=-2 is that 4% more answers were returned within the top 10 documents by the optimal basis (168 vs 159 returned out of 209 total), and the average rank of the returned answers decreased from 1.62 to 1.45 (an average rank

of 1.0 would mean that all answers that were returned within the top 10 documents were returned as the top-ranked document).

5 Related Work and Discussion

There is some confusion over the actual scoring algorithm used in the original paper on LSA by Deerwester et al. [4]. Most likely they used the standard method given by equation (2), i.e. used U_k as the basis set, as does Ding [5]. Berry et al [2] do a careful job explaining the difference between (1) and (2). I interpret [1] as using equation (3), ($U_k \Sigma_k^1$ as basis) for their scoring, but this is quite possibly an incorrect reading.

Kolda and O'Leary [8] (p 328) suggest a scoring algorithm using a "split" Σ_k:

$$score_j = cos(\vec{q}^T U_k \Sigma_k^\alpha, \Sigma_k^{1-\alpha} V_k^T)$$

for various values of a "splitting parameter" α. However they did not seem to interpret this as a basis scaling, and I am unclear on the motivation for the split. This formulation is not equivalent to mine, since it does not add scaling factors of Σ_k^p to the score calculation.

Jiang and Littman [7] present a very interesting formulation and comparison of vector space methods, including LSA. They show that the standard vector space model approximates A by the unitary matrix $\overline{A} = UV^T$, which we might write as

$$\overline{A} = UIV^T = Udiag(1, 1, ...1)V^T$$

while the standard LSA scoring algorithm is equivalent to approximating A by

$$\overline{A}_k = UI_kV^T = Udiag(1, 1, ...1, 0, 0...0)V^T$$

and a third variation, the Generalized Vector Space Model uses the matrix A directly to calculate scores, which of course can be written

$$A = U\Sigma V^T = Udiag(\sigma_1, \sigma_2, ...\sigma_n)V^T.$$

This emphasizes that the difference between these methods is their use of the singular value matrix Σ in the score calculation. The weighted-exponent parameterizations presented in this paper can be formulated in these terms as

$$\overline{A}_k^p = Udiag(\sigma_1^p, \sigma_2^p, ...\sigma_k^p, 0, 0...0)V^T. \quad (5)$$

Their hypothesis of an "ideal singular value plot" deserves more investigation and may shed some light on what value of p is optimal for a dataset.

Ding [5] constructs a statistical model for LSA in which he shows that the rows of U are the maximum likelihood estimates of the "latent semantic structures" that act as the generators of the statistical distribution of documents found in the document set. An important result is that the statistical contribution of $\vec{u}_i$ is proportional to σ_i^2 (modulo a slowly-varying normalization factor). While the details of the normalization factor make things complicated, he is able to derive

the optimal rank for various datasets that is close to the experimentally determined value.

The relationship between optimal rank and optimal value of the weighting exponent, which has been a major feature of the experiments presented here, may contain important insights to the underlying mechanics of LSA. It is certainly noteworthy that the retrieval results converge to a common value for all values of k as the value of p increases (see Figures 2 and 3). As p increases, the factors with larger eigenvalues increase in importance, which probably has the same effect as rank-reduction.

More surprising is the results of Figure 7, in which accuracy is improved by increasing the importance of the factors with smaller eigenvalues (optimal $p < 0$). This effect may have something to do with the correlations between questions and answers in written natural language conversations.

In conclusion, I have defined a family of LSA scoring functions parameterized by a "weighting exponent" p, which weights the score by Σ^p, where Σ is the diagonal matrix of SVD eigenvalues. The usual LSA scoring corresponds to $p = 0$. I have shown experimental evidence on standard test datasets that retrieval precision can often be improved by a few percent using values of p greater than zero. In another experiment, I have shown that values of p less than zero can improve question/answer matching by around 8%.

The optimal value of p is dependent on the dataset, on the SVD approximation rank k, and possibly on other choices of how the term document matrix is constructed, such as term weighting. Therefore the practical consequences of finding the optimal p may turn out to be small. However the theory of how and why the SVD algorithm works for information retrieval is not well understood. It is a challenge to theoretical models of LSA to explain these empirical findings.

Acknowledgments

I would like to thank Dr. Elizabeth Jessup at the University of Colorado, Boulder for pointing out the confusion in LSA scoring algorithms, and for other useful discussions.

Bibliography

[1] Berry, Michael W., Susan T. Dumais and Gavin W. O'Brien *Using Linear Algebra for Intelligent Information Retrieval* in SIAM Review, 37:4 (1995), pp. 573-595. http://www.cs.utk.edu/ library/TechReports/1994/ut-cs-94-270.ps.Z

[2] Berry, M.W., Z. Drmac, and E.R. Jessup *Matrices, Vector Spaces, and Information Retrieval* in SIAM Review 41:2, (1999), pp. 335-362. http://epubs.siam.org/sam-bin/dbq/article/34703

[3] Caron, John *Applying LSA to Online Customer Support: A Trial Study.* Masters Thesis (April 2000). FAQO source code and related documents are at http://www.unidata.ucar.edu/staff/caron/faqo/index.html

[4] Deerwester, S., Dumais, S. T., Landauer, T. K., Furnas, G. W. and Harshman, R. A *Indexing by latent semantic analysis* in Journal of the Society for Information Science, 41(6), 391-407 (1990). http://lsi.research.telcordia.com/lsi/papers/JASIS90.ps

[5] Ding, Chris H.Q. *A Similarity-based Probability Model for Latent Semantic Indexing* in Proc. of 22nd ACM SIGIR 99 Conference, pp.59-65. (August 1999). http://www.nersc.gov/research/SCG/cding/papers_ps/sigir2.ps

[6] Dumais, S. T. *Improving the retrieval of information from external sources* in Behavior Research Methods, Instruments and Computers, 23(2), 229-236 (1991). http://lsi.research.telcordia.com/lsi/papers/BRMIC91.ps

[7] Fan Jiang and Michael L. Littman *Approximate dimension equalization in vector-based information retrieval* in Proceedings of the Seventeenth International Conference on Machine Learning, to appear, (2000) http://www.cs.duke.edu/ mlittman/docs/icml00-lsi.ps

[8] Kolda, Tamara G. and Dianne P. O'Leary *A Semi-Discrete Matrix Decomposition for Latent Semantic Indexing in Information Retrieval* in ACM Transactions on Information Systems, 16 (1998) 322-346. http://www.cs.umd.edu/Dienst/UI/2.0/Describe/ncstrl.umcp/CS-TR-3724

[9] SMART data archives ftp://ftp.cs.cornell.edu/pub/smart/

[10] SVDPACKC Michael Berry, Theresa Do, Gavin O'Brien, Vijay Krishna, and Sowmini Varadhan *SVDPACKC (Version 1.0) User's Guide* Univ. of Tennessee Computer Science Report CS-93-194, revised March 1996. `http://www.netlib.org`